SISTEM KENDALI BERBASIS MIKROKONTROLER MENGGUNAKAN PROTOKOL *MQTT* PADA *SMARTHOME*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Hudan Abdur Rochman
NIM: 135150200111107

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

SISTEM KENDALI BERBASIS MIKROKONTROLER MENGGUNAKAN PROTOKOL *MQTT* PADA *SMARTHOME*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Hudan Abdur Rochman
NIM: 135150200111107

Skripsi ini telah diuji dan dinyatakan lulus pada
04 Mei 2017
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Rakhmadhany Primananda, S.T, M.Kom	Ir.Heru Nurwarsito, M.Kom
NIK: 2016098604061001	NIP: 196504021990021001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 22 Agustus 2017

Hudan Abdur Rochman

NIM: 135150200111107

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, karena berkat rahmat serta bimbingannya penulis dapat menyelesaikan tugas akhir dengan judul "SISTEM KENDALI BERBASIS MIKROKONTROLER MENGGUNAKAN PROTOKOL *MQTT* PADA *SMARTHOME*" dengan baik dan tepat waktu. Penulisan dan penyusunan laporan skripsi ini dapat terlaksana dengan baik karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu diantaranya:

1. Bapak Rakhmadhany Primananda, S.T, M.Kom. selaku dosen pembimbing I yang dalam penulisan ini telah banyak memberi ilmu serta masukan.
2. Bapak Ir.Heru Nurwarsito, M.Kom, selaku dosen pembimbing II yang juga banyak memberi ilmu dan masukan untuk penulisan laporan skripsi ini.
3. Keluarga penulis, khususnya yaitu orang tua penulis Bapak Heru Pribadi dan Ibu Lily Sherly Juhartini, serta kakak dan adik penulis yaitu Harjuno Aryo Dananto, M. Fajar Asqolani, Bagja Fathur Ramadhan, dan Carissa cahya Shalsabila yang telah memberikan dukungan moril dan materil serta motivasi.
4. Nanda Putri Romadhona yang memberikan motivasi lebih untuk terselesaikannya skripsi ini.
5. Hanif Kuncahyo Adi dan Yosia Rimbo Deantama selaku teman sekaligus kakak tingkat yang selalu memberikan bimbingan, arahan, dan serta masukan selama proses pengerjaan skripsi ini.
6. Zulianur Khaqiqiyah selaku teman yang senantiasa membantu dalam dukungan moril serta materil selama pengerjaan skripsi ini.
7. Ryan Iriany, Adi Iman Utama, Teguh Surya, Yogi Suwandy, Malik Abdul Azis, Jefry Kalvin, Artiyan Prasetya, Bayu Laksana Yudha, M. Alfi Fauzan, Fransnesa, dan Dhyono Dhyakso, selaku teman serta sahabat yang selalu membantu dan memberikan motivasi selama pengerjaan skripsi ini.
8. Teman-teman Fakultas Ilmu Komputer, khususnya teman-teman program studi Informatika 2013 dan BEMTIIK periode 3 terima kasih atas segala bantuan dan dukungannya selama ini.
9. Seluruh pihak yang telah membantu secara langsung dan tidak langsung dalam penyelesaian tugas akhir ini yang namanya tidak dapat disebutkan satu per satu oleh penulis.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari sempurna karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Akhirnya semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa Filkom Universitas Brawijaya

Malang, 22 Mei 2017

Penulis
hudanabdurroch@gmail.com

ABSTRAK

Internet of Things (*IoT*) dapat dimanfaatkan untuk melakukan kontroling dan monitoring pada suatu tempat tertentu seperti rumah, yang biasa disebut dengan *smarthome*. Salah satu protokol yang dirasa tepat untuk pengimplementasian *IoT* didalam lingkup *smarthome* adalah protokol *MQTT* yang bertipe *publish/subscribe*. Selain membutuhkan protokol jaringan dibutuhkan juga perangkat-perangkat penunjang lainnya seperti sensor DHT11 yang digunakan untuk membaca kondisi suhu, LDR yang merupakan sensor intensitas cahaya, lampu LED yang akan merepresentasikan *device-device* yang akan di kontrol pada *smarthome* dan mikrokontroler yang berperan sebagai aktuator dari perangkat-perangkat tersebut. Mikrokontroler yang digunakan pada penelitian ini adalah mikrokontroler wemos D1 R2 yang memiliki modul esp8266 yang menunjang pembentukan jaringan secara wireless. pada penilitian ini dibuat contoh sederhana dari *smarthome* dimana 2 buah lampu LED akan mati atau menyala berdasarkan dari nilai data sensor suhu dan cahaya dan dikontrol melalui aplikasi sistem kendali. Selain itu aplikasi dapat menampilkan data sensor suhu dan cahaya dalam bentuk grafik sebagai fungsi monitoring dari sistem kendali. Hasil pegujian menunjukan banyaknya paket yang dikirimkan menggunakan protokol *MQTT* dalam satu waktu mempengaruhi nilai delta time dimana semakin singkat jeda pengiriman waktu semakin kecil nilai delta timenya, dan nilai integritas data yang dikirim dan diterima melalui protokol *MQTT* adalah 100%.

Kata kunci: *Smarthome, MQTT, Mikrokontroler, Wemos*

ABSTRACT

Internet of things (IoT) can be utilized for controlling and monitoring at a particular place such as home, commonly called Smarthome. One of the protocols which is deemed appropriate for the implementation of IoT and the Smarthome is MQTT protocol that has publish/subscribe communication mechanism. In which it requires network protocols, also supporting devices such as a sensor DHT11 which will be used to read the temperature conditions in Smarthome, the LDR which is a sensor of light intensity, LED lights that will represent controlling devices in Smarthome and microcontroller which acts as an actuator. Microcontrollers used in this study is a microcontroller Wemos D1 R2 which has a module ESP8266 that supports the establishment of a wireless network. In this research, a simple prototype of Smarthome is built, where two pieces of LED lights will go on and or out based on the value of temperature and light sensor data, then controlled through the application of the control system. The application can also display temperature and light sensor data in graph form as a function of the monitoring control system. The test result showed the number of packets sent using protocol MQTT at a time affect the delta time, wherein the shorter the sending time the smaller the delta time value, and the integrity value of data which is sent and received through MQTT protocl is 100%.

***Key*words:** *Smarthome, MQTT, Microcontroller, Wemos*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

BAB 1 PENDAHULUAN

1.1 Latar belakang

Internet of things (*IoT*) adalah sebuah konsep dimana perangkat - perangkat elektronik nantinya akan memiliki kemampuan untuk saling berkomunikasi dengan mandiri, saling menerima dan mengirimkan data melalui koneksi jaringan. Salah satu penerapan *internet of things* adalah pada sistem monitoring atau kontroling yang menggunakan sensor dan aktuator pada sebuah lingkungan tertentu seperti *smarthome*. *Smarthome* sendiri merupakan salah satu contoh penerapan *IoT* dalam lingkup yang lebih kecil yaitu rumah. *Smarthome* adalah sebuah istilah yang digunakan untuk menyebut sebuah hunian yang modern dimana terdapat proses – proses otomatis didalam sistem rumah tersebut seperti salah satunya pada sistem pencahayaan rumah tersebut (Naglič & Souvent, 2013). *Smarthome* menghubungkan perangkat-perangkat didalam rumah tersebut menjadi terintegrasi di dalam sebuah sistem yang memiliki pusat kendali untuk melakukuan monitoring maupun kontroling dari perangkat – perangkat di dalam *smarthome* tersebut.

Seiring dengan perkembangan zaman, dalam penerapannya *IoT* maupun *smarthome* dapat di terapkan dengan menggunakan berbagai macam jenis protokol jaringan yang ada. Akan tetapi tentu saja tidak semua protokol baik untuk penerapan *IoT* karena ke tidak tepatan pemilihan protokol untuk menerapkan *IoT* maupun *smarthome* hanya akan membuat sistem menjadi rumit untuk dikembangkan. Oleh karena itu dibutuhkan sebuah protokol yang dapat berkerja dengan efisien dan mudah dalam penerapannya.

Message Queue Telemetry Transport (*MQTT*) adalah sebuah protokol yang dapat di gunakan untuk menerapkan konsep *Internet of things* (*IoT*). *MQTT* dirasa tepat untuk menjadi protokol *IoT* karena *MQTT* bersifat *light weighted message* dan di desain untuk perangkat yang memiliki sumber daya terbatas (Kim, et al., 2015). Dengan sifat - sifat yang dimiliki oleh *MQTT* protokol ini dirasa tepat untuk diterapkan pada suatu sistem kendali *smarthome*. Saat ini penggunaan protokol *MQTT* pada *IoT* lebih banyak digunakan hanya untuk monitoring saja, belum banyak penelitian yang menggunakan protokol *MQTT* untuk melakukan kontroling pada perangkat – perangkat dalam jaringan *IoT.* Namun *MQTT* yang merupakan protokol yang berjalan pada layer aplikasi dan dengan mekanisme protokol *MQTT* yaitu *publish*/*subscribe* yang dapat menyesuaikan pengiriman dan penerimaan pesan dapat digunakan untuk melakukan kontroling dan monitoring sesuai dengan keinginan pengguna, kerena pengiriman dan penerimaan pesan protokol *MQTT* yang berdasarkan topik – topik yang ditentukan.

Maka dari itu dengan paparan masalah diatas dan penjelasan mengenai protokol *MQTT*, melihat adanya kesinambungan permasalahan dengan penjelasan mengenai protokol *MQTT*, maka dalam penelitian ini penulis menggunakan protokol *MQTT* sebagai protokol komunikasi di dalam sistem kendali *smarthome* yang akan dibuat.

Alasan penulis memilih protokol *MQTT* sebagai protokol aplikasi untuk jaringan *IoT* khususnya pada *smarthome* di perkuat dengan penelitan sebelumnya, yaitu *MQTT Based Secured Home Autoation System* (Upadhyay, et al., 2016) dan *Performance Evaluation of MQTT and CoAP via a Common Middleware* (Thangave, et al., 2014). Kedua penelitian di atas menerangkan bahwa protokol *MQTT* menggunakan energi yang sangat sedikit dibandingkan dengan protokol lainnya, dan dapat bekerja dengan baik di dalam lingkungan yang memiliki *bandwidth* rendah dan *latency* tinggi.

Selain membutuhkan teknologi jaringan, agar perangkat – perangkat seperti sensor dan actuator dapat bekerja secara optimal dibutuhkan sebuah alat untuk mengendalikan atau memonitor kondisi dari perangkat – perangkat tersebut. Salah satu perangkat yang dapat di gunakan adalah mikrokontroller yang nantinya akan berfungsi untuk menerima pesan dan mengolah pesan yang akan diterima ataupun yang dikirim oleh perangkat yang terhubung dengannya.

Mikrokontroler adalah sebuah *circuit* elektronik atau mikroprosesor yang telah dilengkapi prosesor, memory, dan antarmuka Input/Output, tidak seperti mikroporsesor yang biasanya hanya memiliki cpu saja. Pada perkembangannya mikrokontroler banyak digunakan untuk membuat sebuah sistem dikarenakan beberapa keunggulan mikrokontroler yaitu ukuran chip yang kecil, cepat, mudah digunakan, dan harganya yang murah (Mohammad, et al., 2009). Salah satu contoh mikrokontroler adalah mikrokontroler Wemos D1 R2 yang merupakan mikrokontroler yang kompetibel dengan arduino dimana Wemos D1 R2 ini telah dilengkapi dengan modul wifi ESP8266 sebagai prosesornya, yang memudahkan untuk pengembangan sistem berbasis *wireless.*

Berdasarkan paparan di atas judul yang di ambil dalam skripsi ini adalah. "Sistem Kendali Berbasis Mikrokontroler Menggunakan Protokol *MQTT* Pada *Smarthome*". Sistem kendali yang dimaksud adalah pengguna dapat melakukan kontroling dan monitoring terhadap *smarthome*. Dimana untuk implementasinya menggunakan lampu LED, sensor suhu yaitu DHT11 dan sensor intensitas cahaya yang disebut Light Dependent Resistor (LDR). Nantinya lampu LED, DHT11 dan LDR akan terpasang pada mikrokontroler Wemos yang berbeda, Dimana DHT11 dan LDR akan mengirimkan data hasil tangkapan dari masing – masing sensor kepada server dan diolah oleh server untuk selanjutnya dilanjutkan kepada lampu LED yang akan melakukan aksi sesuai dengan data yang dikirimkan oleh server, komunikasi di dalam jaringan nantinya menggunakan protokol *MQTT* yang bersifat *lightweight* yang bisa mengirimkan data secara cepat, dan *reliable*. Di harapkan tema yang diambil dalam skripsi ini dapat bermanfaat dalam penelitian di bidang *IoT* khususnya dalam perangkat berbasis mirkontroler khususnya yang menggunakan modul esp866.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dijabarkan, maka dapat dirumuskan beberapa permasalahan sebagai berikut:

1. Bagaimana membuat sistem kendali *smarthome* yang saling terintegerasi?

2. Bagaimana menerapakan protokol *MQTT* untuk komunikasi di dalam sistem kendali *smarthome*?
3. Bagaimana mekanisme pengiriman data antara *publisher*, broker, dan *subscriber*?
4. Bagaimana performa kecepatan dan kehandalan sistem dalam menangani request yang di berikan?

1.3 Tujuan

Tujuan umum penulisan skripsi ini adalah:

1. Dapat membuat sistem kendali *smarthome* yang terintegerasi dengan baik
2. Dapat merancang dan menerapkan protokol komunikasi *MQTT* sebagai metode komunikasi sistem kendali *smarthome* berbasis mikrokontroler.
3. Dapat Merancang komunikasi mengirim dan menerima data dari *publisher* ke broker, dan dari broker ke subcriber.
4. Dapat mengetahui performa kehandalan sistem

1.4 Manfaat

Manfaat yang diperoleh penulis adalah penulis mendapat pengetahuan mengenai protokol komunikasi *MQTT* dalam proses komunikasi antar perangkat. Untuk Program Studi Teknik Informatika, Jurusan Teknik Informatika serta Fakultas Ilmu Komputer, penelitian ini dapat menambah referensi penelitian di bidang jaringan khusunya mengenai protokol *MQTT*, konsep Internet Of Things, dan sistem kendali *smarthome*.

1.5 Batasan masalah

Agar pembahasan tidak melebar dari latar belakang dan terfokus dengan apa yang berkaitan dengan sistem, maka dapat di jabarkan batasan dalam penelitian ini yaitu :

1. Menggunakan mikrokontroler Wemos D1 R2.
2. Implementasi protokol *MQTT* untuk mengontrol perangkat berbasis mikrokontroler wemos.
3. Implementasi protokol *MQTT* untuk memonitoring perangkat berbasis mikrokontroler wemos.
4. Data sensor DHT11 yang digunakan hanya data suhu.
5. Database yang digunakan adalah Redis.
6. Pengguna mengakses sistem menggunakan aplikasi python.

1.6 Sistematika pembahasan

Sistematika pembahasan dari penyusunan penelitian yang direncanakan adalah sebagai berikut:

BAB I PENDAHULUAN

Pendahuluan terdiri dari latar belakang, identifikasi dan pembatasan masalah, rumusan masalah, tujuan dan manfaat serta sistematika pembahasan dari penelitian ini. Selain itu juga sistematika pembahasan digunakan sebagai acuan untuk melakukan pembahasan terhadap penelitian yang dilakukan secara sistematis.

BAB II LANDASAN KEPUSTAKAAN

Bab landasan kepustakaan menjelaskan tentang teori – teori yang di pakai dalam penelitian, temuan dan bahan penelitian yang mungkin sebelumnya diperoleh dari beberapa refrensi yang menunjang penelitian dalam penulisan skripsi. Ada juga teori – teori yang tercakup dalam bab ini adalah mengenai protokol *MQTT*, mikrokontroler wemos, serta Redis.

BAB III METODOLOGI PENELITIAN

Bab Metodologi tahapan-tahapan langkah-langkah dalam melakukan penelitian. Langkah-langkah seperti perancangan, implementasi, pengujian, serta analisa hasil dibahas secara umum. Selain itu pada bab metodologi juga dijelaskan secara singkat mengenai perangkat keras yang dibutuhkan, arsitektur sistem, pengolahan format data serta proses transmisi dalam penelitian.

BAB IV PERANCANGAN

Bab Perancangan menjelaskan tentang bagaimana arsitektur jaringan yang sesuai dengan objek penelitian. Dibuat pula rancangan sistem kendali *smarthome*. Pada bab ini juga akan akan dijelaskan pula bagaimana proses alur data dari *publisher* ke broker, broker ke *subscriber* dan sebaliknya. Skenario pengujian yang mencakup tujuan penelitian juga disampaikan dalam bab ini.

BAB V IMPLEMENTASI

Bab Implementasi menjelaskan tentang bagaimana menerapkan sistem yang telah dibuat berdasarkan sistematika sebelumnya secara rinci beserta dengan langkah-langkah pengerjaan yang dilakukan serta menampilkan gambar-gambar implementasi yang dilakukan.

BAB VI PENGUJIAN DAN ANALISIS HASIL PENGUJIAN

Bab Pengujian dan Analaisis Hasil Pengujian menjelaskan tentang pengujian terhadap sistem yang telah dibuat berdasarkan skenario yang telah dibuat pada bab sebelumnya. Serta menyajikan data hasil pengujian beserta analisis dari implementasi sistem yang telah dibangun. Hasil pengujian didapatkan dari pengujian yang telah dilakukan sesuai dengan skenario pengujian yang telah ditentukan. Dari hasil tersebut dapat dilakukan analisis terhadap kinerja sistem, serta mengetahui bagaimana sistem dapat mencapai tujuan.

BAB VII PENUTUP

Pada Bab Penutup dipaparkan kesimpulan dari pelaksanaan penelitian. Kesimpulan ini dibuat dengan hasil pengujian dan analisis terhadap perancangan

dan implementasi arsitektur sistem sebagai dasarnya. Saran-saran juga diberikan agar hasil dari penelitian ini dapat diperbaiki dan disempurnakan apabila penelitian ini dikembangkan kemudian.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang diperlukan untuk penelitian. Kajian pustaka membahas penelitian yang telah ada dan yang diusulkan. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan, dasar teori meliputi teori protokol komunikasi *MQTT*, mikrokontroller Wemos D1 R2, database redis, dan perangkat - perangkat yang di gunakan.

2.1 Kajian Pustaka

Kajian pustaka akan membahas mengenai penelitian penelitian yang pernah dilakukan sebelumnya yang terkait dan relevan dengan penelitian yang akan di lakukan oleh penulis.

Penelitian pertama adalah penelitian yang dilakukan oleh (Aroon, 2016) pada penelitian ini menggunakan *MQTT* cloud platform untuk kontroling robot secara remote melalui cloud menggunakan platform netpie, pada penelitian ini menunjukan penggunaan platform *MQTT* cloud platform dalam hal ini netpie memiliki waktu transmisi pesan yang kecil, ini dikarenakan kecilnnya paket data yang dikirimkan.

Penelitian kedua adalah penelitian yang di lakukan (Nusantara, et al., 2016) pada penelitian yang berjudul ANALISA METODE *PUBLISH*/*SUBSCRIBE* UNTUK KOMUNIKASI DATA ANTAR PERANGKAT DALAM LINGKUKANGAN *SMARTHOME*, penelitian ini membandingkan protokol *MQTT* dan AMQP, dari penelitian ini disimpulkan *MQTT* lebih cepat dan stabil dibandingkan dengan AMQP.

Penelitian ketiga adalah penelitian yang dilakukan oleh (Kim, et al., 2015) menggunakan *MQTT* untuk komunikasi antar device dengan *IoT* home gateway, pada penelitian ini protokol *MQTT* digunakan karena dapat bekerja dengan baik pada bandwidth yang rendah, dan juga metode *publish*/*subscribe* protokol *MQTT* dirasa tepat untuk sistem home automation. **Tabel 2.1** menjelaskan tentang kajian pustaka yang digunakan dalam penelitian.

Tabel 2.1 Kajian Pustaka

NO	Judul	Penulis	Perbandingan	
			Kajian Pustaka	Tugas Akhir Penulis
1	Study of using *MQTT* Cloud Platform for Remotely Control Robot and GPS Tracking	Nut Aroon	Menggunakan *MQTT* cloud Platform untuk melakukan controlling dan traking pada robot secara remote	Menggunakan protokol *MQTT* dalam komunikasi pertukaran data pada sistem kendali *smarthome* berbasis mikrokontroler

				wemos dengan modul esp8266
2	ANALISA METODE *PUBLISH/SUBSCRIBE* UNTUK KOMUNIKASI DATA ANTAR PERANGKAT DALAM LINGKUKANGAN *SMARTHOME*	M Fikri Nusantara, Sabriansyah Rizqika Akbar, Aditya Rachmadi	Analisa performa protokol *MQTT* pada lingkungan *smarthome*	Implementasi protokol komunikasi *MQTT* untuk komunikasi antar perangkat yang terhubung di dalam sistem
3	*IoT* Home Gateway for Auto-Configuration and Management of *MQTT* Devices.	Seong-Min Kim, Hoan-Suk Choi, Woo-Seop Rhee	Menggunakan protokol *MQTT* untuk komunikasi antar gateway dengan device	Menggunakan protokol *MQTT* untuk komunikasi perangkat dengan database redis

2.2 Dasar Teori

Berdasarkan informasi yang telah didapatkan dari kajian pustaka, maka di dalam "SISTEM KENDALI BERBASIS MIKROKONTROLER MENGGUNAKAN PROTOKOL *MQTT* PADA *SMARTHOME*" terdapat beberapa kajian pustaka yaitu:

2.2.1 *Smarthome*

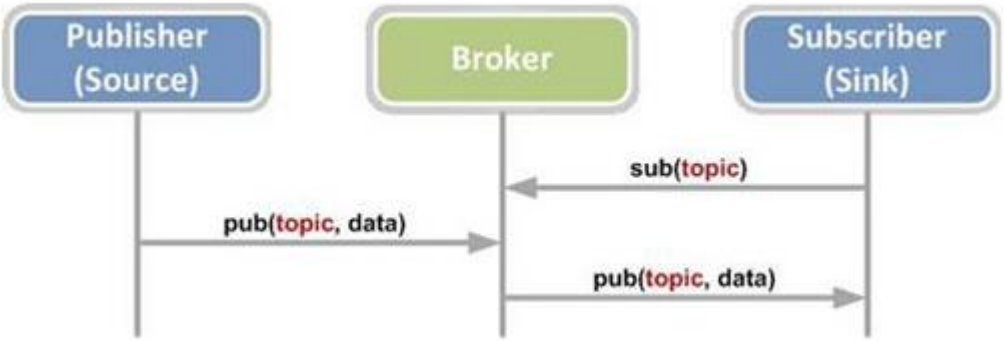
Smarthome atau home automation adalah sebuah konsep hunian dimana di dalam hunian terdapat sebuah sistem yang berjalan secara otomatis untuk meningkatkan kenyamanan dan keamanan penghuni hunian, *smarthome* merepresentasikan hunian yang memiliki sistem otomatis yang canggih untuk sistem pencahayaan, sistem keamanan, sistem pengaturan suhu, *home energy management,* sistem multimedia, dan lainnya. *Smarthome* menghubungkan sistem - sistem otomatis tersebut kedalam satu jaringan yang terintegerasi dengan sebuah pusat kontrol, salah satu tujuan dari *smarthome* adalah menyesuaikan kebutuhan dari penghuni sesuai dengan standar kenyamanan masing - masing dari penghuni hunian tersebut.

Melalui sistem kontrol penghuni dapat mengatur sistem – sistem yang terhubung sesuai dengan kebutuhan penghuni, perangkat - perangkat yang ada di dalam *smarthome* dapat terhubung dengan menggunakan berbagai cara salah satunya menggunakan tekonologi komunikasi *wireless* seperti *WiFi*. Dalam

pengimplementasian *smarthome* dibutuhkan device – device untuk menunjang seperti perangkat sensor yang akan menjadi pengumpul data dan informasi untuk *smarthome*, aktuator yang akan menjadi pemicu device – device yang terhubung di dalam sistem, dan aplikasi sistem kontrol yang akan menjadi perantara penghuni dengan sistem *smarthome*. Aktuator sendiri dapat berupa mikrokontroler, selain itu penggunaan mikrokontroler dapat digunakan untuk kebutuhan pembuatan jaringan di dalam *smarthome* itu sendiri.

2.2.2 Message Queue Telemetry Transport (*MQTT*)

Message Queue Telemetry Transport (MQTT) adalah sebuah protokol komunikasi data *machine to machine* (M2M) yang berada pada layer aplikasi. *MQTT* bersifat lightweight message artinya *MQTT* berkomunikasi dengan mengirimkan data pesan yang memiliki header berukuran kecil yaitu hanya sebesar 2bytes untuk setiap jenis data, sehingga dapat bekerja di dalam lingkungan yang terbatas sumberdayanya seperti kecilnya bandwidth dan terbatasnya sumberdaya listrik. Protokol *MQTT* juga menjamin terkiriminya semua pesan walaupun koneksi terputus sementara. Protokol *MQTT* menggunakan metode *publish*/*subscribe* untuk metode komunikasinya, cara kerja protokol *MQTT* dapat dilihat seperti pada gambar **Gambar 2.1** beriikut:



Gambar 2.1 Cara Kerja *MQTT*

[Sumber tessel.io]

Publish/*subscribe* sendiri adalah sebuah pola pertukaran pesan di dalam komunikasi jaringan dimana pengirim data disebut *publisher* dan penerima data disebut dengan *subscriber*. Metode *publish*/*subscribe* adalah salah satu tipe komunikasi *indirect* atau tidak langsung yang memiliki beberapa kelebihan salah satunya yaitu *loose coupling* atau *decouple* dimana berarti antara produsen data dan konsumen data tidak saling terhubung secara langsung. Terdapat 3 buah *decoupling* yaitu *time decoupling*, *space decoupling* dan *synchronization decoupling*. *Time decoupling* adalah sebuah kondisi dimana *publisher* dan *subscriber* tidak harus saling aktif pada waktu yang sama. *Space decoupling* adalah dimana *publisher* dan *subscriber* aktif di waktu yang sama akan tetapi antara *publisher* dan *subscriber* tidak saling mengetahui keberadaan dan identitas satu

sama lain. Dan yang terakhir adalah *synchronization decoupling* kondisi dimana pengaturan event baik itu penerimaan atau pengiriman pesan di sebuah node hingga tidak saling mengganggu satu sama lain (Adi, et al., 2016). Dengan kelebihan – kelebihan tersebut jika terjadi kerusakan disalah satu node yang terhubung dalam jaringan menggunakan *MQTT* tidak akan terlalu berpengaruh pada node – node lainnya sehingga jika terdapat kerukasan pada satu node *publisher* atau *subscriber*, node – node yang lain akan tetap berfungsi sehingga memudahkan untuk melakukan pengcekan jika terjadi kerusakan.

Pengiriman data pada *MQTT* didasari oleh topik, topik ini nantinya yang akan menentukan pesan dari *publisher* harus dikirim pada *subscriber* yang mana. Topik ini dapat bersifat hirarki, dimana *MQTT* topik memiliki tipe data string dan untuk perbedaan hirarki atau level dari topik digunakan tanda baca "/", pada *MQTT* topic dikenal 2 buah *wildcard* karakter untuk subscription topik yaitu "+" dan "#", penggunaan *wildcard* karakter "+" adalah sebuah *single level wildcard* yang digunakan untuk mencocokkan topik yang di *subscriber* dengan apapun, contohnya seperti rumah /+/ cahaya, adalah sama dengan rumah/taman/cahaya, atau rumah/kamar/cahaya, tetapi tidak sama dengan rumah/taman, karena karakter "+" adalah single wildcard karakter yang berati mencocokkan topik dengan semuanya pada level tertentu, sedangkan karakter "#" adalah *multi-level wildcard* karakter dimana karakter ini akan mencocokkan topik pada setiap level karakter ini ada dan setelahnya, contoh rumah/# berarti sama dengan rumah/taman, atau rumah/taman/cahaya atau rumah/cahaya (Solace, n.d.).

Broker merupakan komponen yang paling penting di dalam arsitektur *publish*/subscribe dan *MQTT*. Broker adalah sebagai perantara pertukaran pesan antara *publisher* dan *subscriber*. Masing - masing dari *publisher* dan *subscriber* harus terkoneksi ke broker untuk mengirimkan ataupun menerima pesan, broker akan menerima seluruh pesan yang di *publish* oleh *publisher*, untuk selanjutnya diteruskan kepada *subscriber* berdasarkan dengan topik yang sesuai dengan pesan yang dikirim dan topik yang di *subscribe* oleh *subscriber*, ketika *subscriber* tidak aktif maka broker akan menyimpan pesan pada *buffer* untuk selanjutnya dikirimkan ketika *subscriber* telah aktif kembali.

MQTT memiliki 3 level *quality of service* (QoS) dalam pengiriman pesannya yaitu 0,1,2. Pada level 0 atau disebut dengan "*at most once delivery message"* QoS level 0 ini adalah QoS paling cepat untuk mengirim pesan, QoS ini akan menjamin mencoba mengirim dengan usaha terbaiknya, pesan akan langsung dikirimkan tanpa menunggu *acknowledge* dari sisi penerima. Namun pengiriman dengan menggunakan QoS level 0 memungkinkan hilangnya pesan jika penerima terputus secara tiba – tiba. QoS level 1 atau disebut juga "*at least once*" dengan opsi level ini *MQTT* akan mengirimkan pesan minimal sekali. Pesan yang dikirimkan nantinya akan di acknowledge oleh penerima, kemudian *MQTT* akan mengirimkan kembali pesan yang menggunakan QoS level 1 jika pengirim tidak menerima acknowledge dari penerima. Pesan yang dikirimkan menggunakan QoS level 1 juga akan di simpan oleh broker pada database. Untuk QoS level 2 atau "*exactly once delivery*" QoS ini akan menjami pesan dikirim dan diterima pada sisi penerima. QoS ini adalah yang paling aman namun juga paling lambat dibandingkan dengan 2 level

QoS lainnya, jaminan pesan akan sampai dikarenakan adanya komunikasi 2 arah dari pengirim dan penerima (Lampkin, et al., 2012).

2.2.2.1 Mosquitto Broker

Mosquitto broker adalah salah satu *open source* broker pesan yang mengimplementasikan protokol *MQTT* versi 3.1 dan 3.1.1. Broker mosquitto juga mendukung implementasi server *lightweight* dari *MQTT* maupun *MQTT*-SN, mosquitto broker ditulis dalam bahasa pemrogram C dengan alasan agar dapat tetap bekerja pada mesin yang tidak mendukung JVM. Dari hasil pengujian yang telah dilakukan sebelumnya broker mosquitto dapat mendukung 100.000 koneksi secara bersamaan (Eclipse, 2013).

2.2.2.2 Paho *MQTT*

Paho *MQTT* adalah sebuah library untuk *MQTT* client yang dikembangkan oleh eclipse, paho library tersedia untuk berbagai bahasa pemrograman seperti Lua, Python, C++ dan Javascript. Paho *MQTT* mendukung penggunakan 3 level QoS *MQTT* untuk pengriman/*publish* pesan dan juga untuk penerimaan/*subscribe* pesan (Eclipse, 2011).

2.2.3 Mikrokontroler

Mikrokontroler adalah sebuah papan rangkaian elektronik yang didalamnya sudah terdapat *chip* mikrokontroler atau cpu kecil, memory dan interface Input/Output, mikrokontroller banyak digunakan untuk embeded sistem dan komunikasi *machine to machine* (M2M) memiliki beberapa keunggulan seperti hemat sumber daya, fleksibel dan harganya yang murah.

2.2.3.1 Wemos D1 R2

Wemos D1 R2 adalah sebuah mikrokontroler yang kompatibel/mirip dengan arduino uno hanya saja wemos D1 R2 berbasis modul ESP8266-12, bahasa pemrograman yang digunakan untuk memprogram wemos D1R2 ini adalah bahasa pemrograman C namun modul esp8266 sudah memiliki cukup banyak library untuk digunakan sehingga pemrograman mikrokontroler berbasis modul esp8266 menjadi relatif mudah meskipun untuk pemula, untuk melakukan pemrograman pada board Wemos D1 R2 ini dapat menggunakan aplikasi Arduino IDE, wemos D1 R2 memiliki 11 digital input/output pins, 1 analog input pin, *micro-usb* untuk koneksi, dan power jack 9-24V daya input (Wemos, n.d.), bentuk fisik Wemos D1 R2 ditunjukan pada **Gambar 2.2** berikut:

Gambar 2.2 Bentuk Fisik Wemos D1 R2

[sumber : instructables.com]

Wemos D1 R2 dengan modul esp-8266 memiliki spesifikasi teknis seperti yang diterangkan pada **Tabel 2.2** berikut:

Tabel 2.2 Spesifikasi Teknis Wemos D1 R2 Esp8266

Microkontroler	**ESP-8266**
Daya Operasi	3.3V
Digital I/O pins	11
Analog Input pins	1(max input 3.2V)
Kecepatan Clock	80MHz/160MHz
Flash	4M bytes
Panjang	68.6mm
Lebar	53.4 mm
Berat	25g

Semua pin pada wemos D1 R2 bekerja pada daya 3.3V untuk spesifikasi dari pin yang dimiliki oleh Wemos D1 R2 ini diterangkan pada **Tabel 2.3** berikut:

Tabel 2.3 Tabel Pins Wemos D1 R2 Esp8266

PIN	**Function**	**ESP-8266 Pin**
TX	TXD	TXD
RX	RXD	RXD
A0	Analog Input	A0
D0	IO	GPIO16
D1	IO,SCL	GPIO5
D2	IO,SDA	GPIO4
D3	IO,10K pull-up	GP1O0

D4	IO,10K pull-up,Builtin_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO,MISO	GPIO12
D7	IO,MOSI	GPIO13
D8	IO, 10k Pulldown SS	GPIO15
G	Ground	GND
5V	5V	5V
3V3	3.3V	3.3V
RST	Reset	RST

2.2.3.2 Esp8266

Esp8266 adalah modul wifi yang terintegerasi dengan protokol TCP/IP, dan memiliki kapabilitas sebagai *micro processor unit* (MCU), yang berarti esp8266 ini memberikan kemampuan kepada semua mikrokontroler untuk mengakses jaringan WIFI yang tersedia. Modul Esp8266 cukup kuat untuk melakukan proses dan kapabilitas penyimpanan datanya, yang memungkinkan modul ini untuk terintegerasi dengan sensor ataupun aplikasi lain yang spesifik melalu GPIOs miliknya, versi terbaru dari ESP8266 wifi modul memiliki peningkatan ukuran flash disk dari 512KB menjadi 1MB. Esp8266 memiliki fitur sebagai berikut (sparkfun, n.d.):

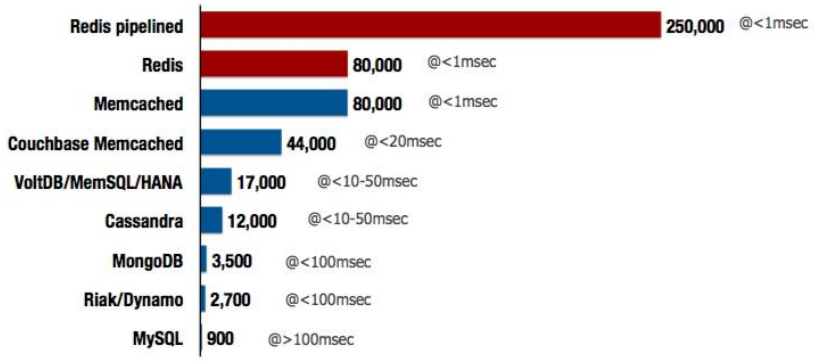
- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLLs, regulators, DCXO and power management units
- +19.5dBm output power in 802.11b mode
- Power down leakage current of <10uA
- 1MB Flash Memory
- Integrated low power 32-bit CPU could be used as application processor
- SDIO 1.1 / 2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval

- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)

2.2.4 Redis

Remote Dictionary Server (Redis) adalah database tipe database *nosql* yang bersifat *key-value* dan penyimpanan datanya di dalam memori, redis biasanya digunakan sebagai database, cache, maupun broker redis mendukung berbagai jenis data struktur seperti string, hashes, lists, sets, sorted sets dengan query, bitmaps, hyperloglogs, dan geospatial index.

Untuk dapat meraih performa yang bagus redis bekerja dengan memory dataset, bergantung dengan berbagai kasus penggunaan redis juga dapat membackup datanya pada disk sesuai dengan pengaturan yang dilakukan, Redis sendi memiliki konsep yang mirip dengan database – database lainnya akan tetapi berbeda seperti database yang lain, yang pada umumnya data ditampilkan pada sebuah aplikasi secara bersamaan, redis memerlukan perintah pada *command line interface* (CLI) nya untuk menampilkan datanya, redis sendiri mendukung fitur expire sehingga dalam penggunaannya tidak perlu khawatir untuk kehabisan memori, kecepatan performa dari redis dapat di lihat pada **Gambar 2.3** berikut:



Gambar 2.3 Perbandingan Performa Redis dengan Database Lain

[sumber : redislabs.com]

2.2.4.1 NoSql

Nosql adalah database management system yang lebih sederhana dibandingkan dengan database relasional biasanya, dan nosql menyediakan tingkat skalabilitas dan ketersedian yang tinggi (Klein, et al., 2015). Nosql sendiri sering berarti tidak menggunakan table dan sql didalam pembentukan databasenya, dan biasanya database nosql menggunakan mekanisme *key-value* untuk pembenetukan databasenya.

2.2.5 Light Emitting Diode (LED)

Light Emitting Diode atau lebih dikenal denga LED ataupun lampu LED adalah sebuah jenis lampu semikonduktor yang memiliki dua buah meterial semikonduktor yaitu semikonduktor postif (*p*) yang disebut dengan anode dan

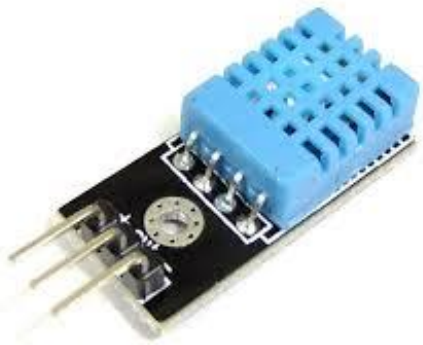
semikonduktor negatif (*n*) yang disebut cathode, LED akan memancarkan cahaya jika diberikan tegangan yang cocok. Cahaya yang dipancarkan merupakan cahaya monokromatik yang tidak koheren, warna yang dihasilkan oleh LED ditentukan juga oleh konduktor yang digunakan. Bentuk lampu LED yang digunakan dapat dilihat seperti pada **Gambar 2.4** berikut:

Gambar 2.4 Gambar Bentuk Lampu LED

[Sumber : ultraleds.co.uk]

2.2.6 Sensor DHT 11

DHT 11 merupakan salah satu sensor suhu dan kelembaban yang menghasilkan output digital, DHT 11 menjamin reliabilitas dan kestabilan yang tinggi, pada penelitian ini hanya akan digunakan nilai suhu dari hasil tangkapan sensor DHT11, DHT 11 sendiri dapat mengukur suhu dengan skala dari 0-50^{O} C, memiliki ukuran yang kecil DHT 11 juga tidak mengonsumsi banyak daya sehingga sangat cocok digunakan untuk mikrokontroller. Bentuk fisik dari DHT 11 dapat dilihat pada **Gambar 2.5** berikut:



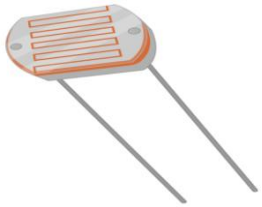
Gambar 2.5 Bentuk Fisik DHT 11

[Sumber : http://www.jualarduino.com/dht11]

2.2.7 Light Dependent Resistor (LDR)

Light Dependent Resistor (LDR) adalah sebuah resistor yang dimana nilai resistornya dipengaruhi oleh intensitas cahaya yang diterima oleh LDR, nilai hambatan dari LDR akan turun sangat rendah ketika cahaya yang diterima oleh LDR sangat terang sedangkan jika LDR berada didalam tempat gelap maka nilai hambatannya akan tinggi, dengn sifatnya yang seperti itu LDR dapat digunakan sebagai sensor cahaya, LDR sendiri terbuat dari sebuah semikonduktor yang tidak

terlindung dari sinar cahaya sehingga ketika LDR tepapar intensitas cahaya yang tinggi maka LDR akan melepaskan elektron bebas sehingga dapat menghasilkan listrik.Bentuk fisik LDR dapat dilihat pada **Gambar 2.6** berikut:



Gambar 2.6 Bentuk Fisik LDR

[Sumber : raspberrypi.org]

BAB 3 METODOLOGI

3.1 Jenis Penelitian

Jenis penelitian yang dilakukan oleh penulis adalah penelitian implementatif dan pengembangan. Penelitian jenis ini dalam rangka membuat sebuah perangkat lunak dan sebuah sistem, penelitian dimulai dengan analisa kebutuhan, pembuatan perangkat lunak dan sistem, serta pengujian produk perangkat lunak dan sistem.

3.2 Metodologi Penelitian

Metodologi penelitian yang akan dilakukan pada penelitian ini secara umum ditunjukkan pada **Gambar 3.1** di bawah ini.

Gambar 3.1 Metode Penelitian

3.2.1 Studi literatur

Studi literatur dilakukan bertujuan untuk mempelajari serta memahami konsep-konsep sistem agar ketika dilakukan perancangan tidak terlalu mengalami kendala. Jenis literatur yang digunakan adalah artikel jurnal. Adapun yang perlu menjadi bahan studi literatur pada penelitian ini meliputi :

- *Smarthome*
- *MQTT*
- Mikrokontroler
- Redis

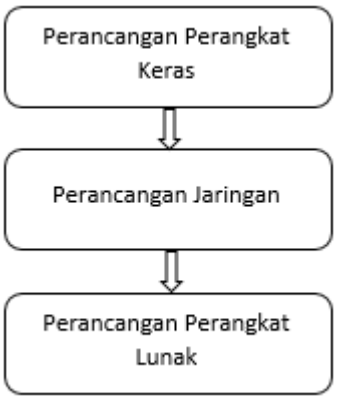
- Lampu LED
- Sensor DHT 11
- Light Dependent Resistor

3.2.2 Analisis Kebutuhan

Pada tahap ini merupakan tahap menganalisa kebutuhan sistem yang akan digunakan. Aplikasi ini menggunakan dua komponen yaitu komponen *hardware* dan komponen *software*. Analisis Kebutuhan sistem diperlukan agar dapat mengetahui hal-hal yang diperlukan dalam pengembangan sistem untuk menghindari penggunaan sumberdaya yang tidak perlu, analisis kebutuhan juga digunakan untuk acuan dalam merancang sistem, sehingga perancangan nantinya dapat terbentuk secara sistematis dan terarah. Dalam melakukan analisis kebutuhan salah satu cara yang dapat digunakan adalah melalui kepustakaan yang telah ada, kepustakaan berlandaskan pada penelitian-penelitian sejenis yang telah dilakukan untuk mengetahui perangkat apa saja yang dapat digunakan dalam pengembangan sistem yang akan dibuat selanjutnya.

3.2.3 Perancangan

Perancangan dilakukan agar penelitian dapat berjalan dengan baik dan sistematis. Perancangan bertujuan untuk memberikan gambaran mengenai implementasi perangkat keras, perancangan jaringan, perangkat lunak dari penelitian yang akan dilakukan, alur perancangan secara umum dapat dilihat pada **Gambar 3.2** berikut:



Gambar 3.2 Perancangan

Pada perancangan perangkat keras akan digambarkan bagaimana rancangan rangkaian antar perangkat yaitu, antara semsor mikrokontroler, sensor DHT11, sensor cahaya, dan lampu LED, perancangan perangkat keras juga akan menggambarkan bagaimana masing-masing sensor dan lampu LED dapat terintegerasi dengan mikrokontrolernya masing-masing, perancangan perangkat keras sendiri teridiri dari, perancngan integerasi DHT11 dengan mikrokontroler, perancangan sensor LDR dengan mikrokontroler, dan perancangan Integerasi Lampu LED dengan mikrokontroler.

Pada perancangan jaringan akan digambarkan bagaimana arsitektur jaringan antar perangkat nantinya, dimana data dari sensor nantinya akan dapat dikirimkan oleh mikrokontroler menuju broker, dan lampu LED dapat menerima data dari komputer server melalui mikrokontroler untuk melakukan aksi sesuai dengan data yang telah diterima oleh komputer server dari broker, pada perancangan jaringan terdiri perancangan aristektur jaringan, perancangan koneksi sistem dengan mosquitto broker, dan perancangan peran *publish*/*subscribe* pada sistem.

Pada perancangan perangkat lunak akan dirancangan bagaimana sistem kendali yang dimaksud, perancangan ini akan menggambarkan komunikasi antar perangkat menggunakan protokol *MQTT*, pada perancangan perangkat lunak juga akan digambarkan bagaimana monitoring dan kontroling terhadap sistem yang dibuat yang nantinya akan ditampilkan pada komputer server, perancangan perangkat lunak sendiri terdiri dari, perancangan aplikasi sistem kendali, perancangan database redis, perancangan monitoring suhu dan intensitas cahaya dari sensor, perancangan kontroling Lampu LED, perancangan monitoring kondisi lampu LED.

3.2.4 Implementasi

Implementasi sistem dilaksanakan berdasarkan perancangan yang telah dibuat sebelumnya. Untuk melakukan implementasi sistem, ada beberapa tahapan yaitu:

1. Implementasi perangkat keras berbasis mikrokontroler pada tahap ini akan dilakukan konfigurasi perangkat sensor dan lampu LED dengan mikrokontroler wemos D1 R2. Konfigurasi ini dilakukan untuk memastikan bahwa semua perangkat dapat terintegrasi dengan mikrokontroler wemos D1 R2. Implementasi perangkat keras terdiri dari Implementasi integrasi DHT11 dengan mikrokontroler, implementasi integrasi sensor LDR dengan mikrokontroler, implementasi lampu LED dengan mikrokontroler
2. Implementasi jaringan pada tahap ini akan diimplementasikan menganai bagaimana setiap perangkat terhubung pada akses point, koneksi perangkat pada mosquitto broker, dan peran *publish*/*subscribe* antar masing masing perangkat. Implementasi arsitektur jaringan, implementasi koneksi sistem dengan mosquitto broker, dan implementasi peran *publish*/*subscribe* pada sistem.
3. Implementasi perangkat lunak, pengimplementasian perangkat lunak meliputi beberapa implementasi sebagai berikut :
    - Implementasi aplikasi sistem kendali. Aplikasi sistem kendali digunakan sebagai antar muka bagi pengguna untuk menjalankan sistem. Aplikasi ini digunakan untuk melakukan kendali terhadap *smarthome*, kendali yang dimaksud adalah bagaimana pengguna dapat melakukan kontroling dan monitoring terhadap *smarthome*.

- Implementasi database Redis. Tahap ini adalah membuat sebuah database yang tersimpan di dalam memori, dimana database menyimpan data-data yang didapatkan dari hasil komunikasi antara mosquitto broker dengan sensor.
- Implementasi monitoring suhu dan intensitas cahaya. pengimplementasian ini adalah menampilkan semua data yang didapatkan oleh sensor dan menampilkan kondisi/state lampu LED pada aplikasi client, sehingga pengguna dapat membaca informasi dengan mudah.
- Implementasi kontroling lampu LED, implementasi kontrolling lampu LED adalah implementasi untuk nyala/matinya lampu LED berdasarkan dari nilai data sensor yang dapat maupun secara kontrol manual melalui aplikasi sistem kendali.
- Implementasi montitoring status kondisi lampu LED, pada implementasi ini dilakukan untuk menampilkan kondisi/state lampu LED yang terintegerasi dalam sistem *smarthome*.

3.2.5 Pengujian dan Analisis Hasil Pengujian

Pengujian sistem dilakukan untuk mengetahui keberhasilan dan kekurangan yang dimiliki oleh sistem yang telah dibuat. Pengujian sistem juga dilakukan untuk mengetahui tingkat keandalan sistem dalam mengirimkan data ke client, sehingga dapat diketahui juga keberhasilan pengiriman data dari sumber ke tujuan. Beberapa skenario yang dilakukan untuk pengujian adalah sebagai berikut.

1. Pengujian sistem kendali *smarthome* dapat saling terintegerasi dan dapat melakukan monitoring serta kontroling pada *smarthome*.
2. Pengujian keberhasilan penerapan protokol komunikasi *MQTT* pada sistem.
3. Pengujian komunikasi pengiriman data antara *publisher* ke broker, dan broker ke *subscriber*.
4. Pengujian kehandalan sistem dalam menerima request dan lalulintas data.

Sementara analisa hasil pengujian dilakukan untuk mengetahui apakah sistem yang dibuat telah sesuai dengan tujuan perancangan sistem tersebut. Terdapat pengujian yaitu, pengujian delay yang terjadi didalam sistem dalam menangani request, pengujian paket lost yang terjadi dalam setiap komunikasi yang terjadi didalam sistem. Pengujian ini dilakukan dengan beberapa skenario.

3.2.6 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dan saran merupakan tahapan akhir dari proses penelitian, yang nantinya dapat disimpulkan mengenai bagaimana sistem yang telah dibuat, kesimpulan dapat diambil dari kelebihan serta kekurangan sistem yang telah dibuat, dan kesusaian sistem antara teori dan praktik, yang nantinya

akan menjawab rumusan masalah yang telah dirumuskan sebelumnya. Saran dimaksud untuk memebrikan masukan terhadap kekurangan – kekurangan yang ada di dalam sistem yang telah dibuat, yang nantinya dapat sebagai pertimbangan untuk penelitian selanjutnya.

BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

Analisis kebutuhan dan perancangan diperlukan agar dalam pengembangan sistem dapat berjalan dengan sistematis, analisis kebutuhan berisi mengenai perangkat – perangkat dan kebutuhan lainnya yang meliputi kebutuhan perangakat kerasa, dan perangkat lunak. Sementara perancangan digunakan sebagai acuan dari implementasi sehingga pengimplemenatasian dapat terarah dan sesuai, perancangan terdiri dari perancangan perangkat keras yang meliputi bagaiamana integrasi antar perangkat dan arsitektur jaringan, dan perancangan perangkat lunak dimana akan digambarkan alur komunikasi pertukaran data antar perangkat menggunakan protokol komunikasi *MQTT*, dan proses pengolahan pesan serta proses penyimpanan pesan pada database.

4.1 Analisis Kebutuhan

Pada tahap ini merupakan tahap menganalisa kebutuhan sistem yang akan digunakan, seperti yang telah di jelaskan pada bab sebelumnya *smarthome* menggabungkan beberapa sistem yang ada kedalam sebuah jaringan tersendiri yang dapat dikendalikan melalu sebuah pusat kontrol. Pada penelitian ini akan dibuat sebuah simulasi *smarthome* yang memiliki 2 buah sistem otomatis yaitu sistem pencahayaan otomatis dan sistem pengaturan suhu ruangan otomatis yang nantinya dapat dikendalikan melalu sebuah aplikasi sistem kendali. *Smarthome* yang dibuat membutuhkan dua komponen yaitu komponen hardware dan komponen software, kebutuhan sistem terdiri dari:

a. Perangkat Keras
    1. Mikrokontroller Wemos D1 R2, Mikrokontroler Wemos D1 R2 digunakan sebagai aktuator dalam proses pengambilan data sensor, eksekusi perintah yang diterima, pengiriman data, serta pembuatan infrastruktur jaringan.
    2. Lampu LED berwana biru yang merepresentasikan pendingin ruangan pada hunian, yang selanjutnya akan disebut dengan LED 1.
    3. Lampu LED berwarna biru yang merepresentasikan lampu untuk pencahayaan pada hunian, yang selanjutnya akan disebut dengan LED 2.
    4. Sensor DHT11
    5. Light Dependent Resistor.
    6. Komputer / Laptop digunakan sebagai server sistem.
b. Perangkat Lunak
    1. Program Mikrokontroller

    Program mikrokontroller harus dapat mengeksekusi perintah dan dapat membaca serta mengirimkan data yang didapat dari sensor.

Program perangkat sensor dituliskan dalam bahasa C dan dijalankan di Arduino IDE

2. Program Aplikasi Sistem Kendali

Program server harus dapat menjadi penjembatan antara perangkat sensor dan client. Selain sebagai lalu lintas komunikasi, server juga diharapkan dapat mengatur database dalam bentuk Redis. Selain itu juga sistem server digunakan sebagai sarana untuk pengguna melakukan kontroling dan monitoring pada sistem. Program aplikasi server dituliskan dalam bahasa python

3. Database Redis

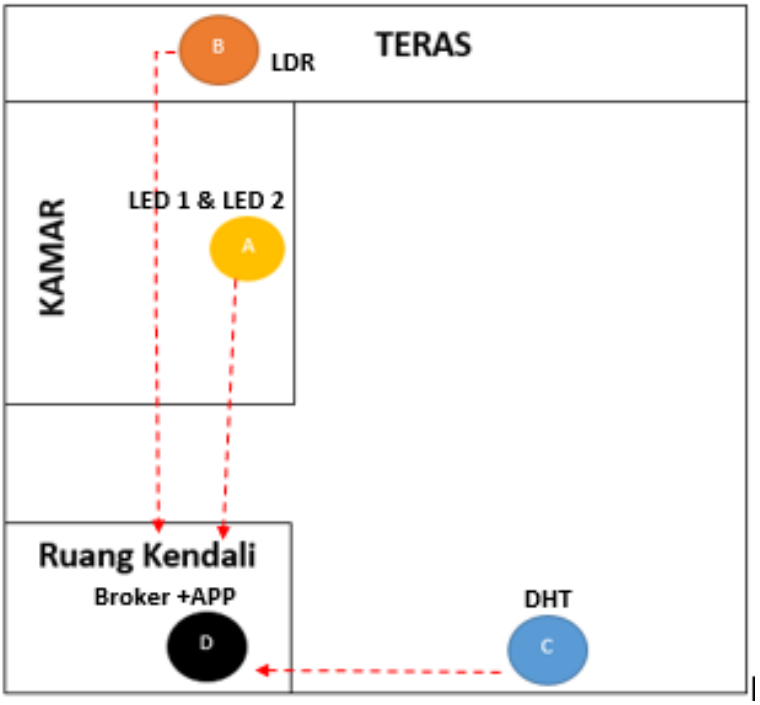
Database Redis digunakan untuk menyimpan semua hasil data yang siap dikirimkan ke aplikasi sistem kendali.

4. Mosquitto Broker

Mosquitto broker digunakan sebagai broker dalam implementasi protokol *MQTT* yang akan digunakan.

4.2 Perancangan

Pada penelitian ini akan dibuat sebuah contoh sederhana mengenai *smarthome* dimana akan terdapat 2 buah sistem didalamnya yaitu sistem pencahayaan otomatis dan sistem pengaturan suhu otomatis yang dapat dikendalikan dari sebuah aplikasi sistem kendali Sistem pengaturan suhu ruangan otomatis yang dimaksud akan menyalakan atau mematikan lampu LED 1 yang merepresentasikan pendingin ruangan beradasarkan dengan nilai suhu yang dibandingkan dengan nilai parameter yang ditentukan oleh pengguna. Sedangkan sistem pencahayaan otomatis yang dimaksud adalah lampu LED 2 yang merepresentasikan lampu untuk pencahyaan rumah akan menyala atau mati secara otomatis sesuai dengan nilai sensor intensitas cahaya yang dibandingkan dengan nilai parameter. Pengguna juga nantinya dapat menyalakan atau mematikan lampu LED 1 dan lampu LED 2 secara manual melalui aplikasi sistem kendali. Penggambaran topologi *smarthome* yang dibuat pada penelitian ini dapat dilihat pada gambar **Gambar 4.1** berikut:



Gambar 4.1 Topologi *Smarthome*

Pada penelitian ini akan mengatur suhu dan pencahayaan pada ruangan kamar pada sebuah hunian. Pada lingkaran A adalah tempat mikrontroler wemos yang terintegerasi dengan LED 1 dan LED 2, lingkaran B adalah mikrokontroler yang terintegerasi dengan LDR, lingkaran C adalah mikrokontroler yang terintegerasi dengan sensor suhu, dan lingkaran D adalah komputer server yang memiliki aplikasi sistem kendali. Penempatan sensor LDR berada di teras dikarenakan sensor akan mengukur cahaya yang masuk kedalam ruangan sehingga sensor perlu ditempatkan ditempat masuknya cahaya, sedangkan penempatan sensor DHT11 di ruangan berbeda dengan LED 1 yang merepresentasikan pendingin ruangan dikarenakan sensor DHT11 akan mengukur nilai suhu disekitar ruangan yang terdapat LED 1 sebagai acuan suhu di dalam ruangan yang terdapat LED 1.

Setelah penempatan node ditempatkan sesuai dengan topologi kemudian semua node akan terhubung pada satu akses point yang sama yaitu node D yang akan bertindak sebagai akses point, garis putus – putus berwarna merah menunjukan koneksi antar node ke akses point, dan koneksi dengan mosquitto broker yang berada pada node D.

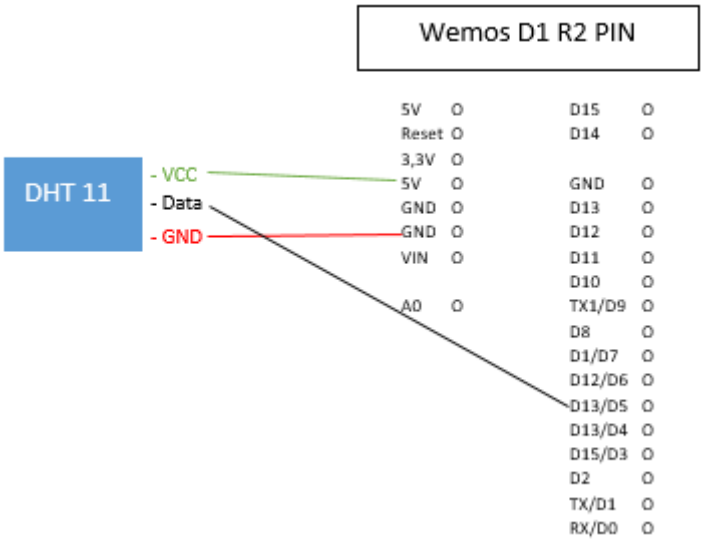
4.2.1 Perancangan Perangkat Keras

Sebelum menghubungkan antar perangkat menjadi sebuah sistem, terlebih dulu diperlukan masing – masing mikrokontroler terintegerasi dengan sensor dan lampu LED, mikrontroler dibutuhkan sebagai sumber daya dan sarana pengirim pesan bagi sensor, dan sarana penerima pesan bagi lampu LED untuk menjalankan perintah. Integerasi antara sensor LDR dengan mikrokontroler wemos D1 R2 berbeda dengan integerasi antara sensor DHT11 dengan mikorkontroler wemos

D1 R2, integerasi antara lampu LED dengan mikrokontroler wemos D1 R2 pun berbeda, maka dari itu dibutuhkan perancangan perangkat keras.

4.2.1.1 Perancangan Integerasi DHT11 dengan Mikrokontroler

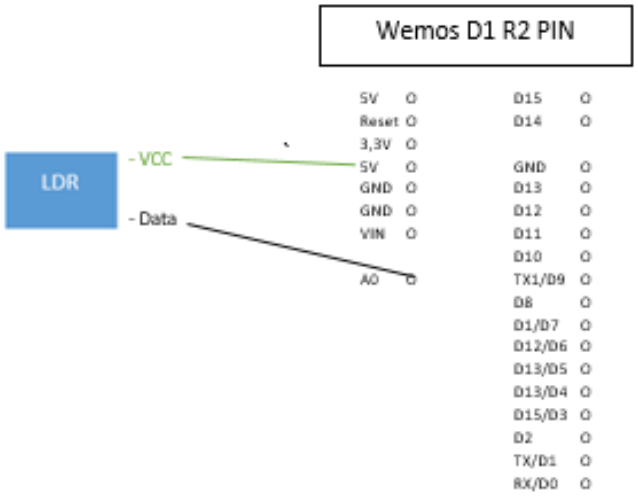
DHT 11 memiliki 3 buah pin yang terdiri dari VCC, GND, dan data, dimana pin VCC harus disambungkan dengan pin power pada mikrokontroler, pin GND harus disambungkan dengan pin ground pada mikrokontroler, dan pin data harus disambungkan pada salah satu pin GPIO pada mikrokontroler wemos D1R2, rancangan kofigurasi untuk mengintegerasikan DHT11 dengan mikrokontroler wemos dapat dilihat pada **Gambar 4.2** berikut:



Gambar 4.2 Integerasi DHT 11 Dengan Mikrokontroler Wemos D1 R2

4.2.1.2 Perancangan Integerasi Sensor LDR dengan Mikrokontroler

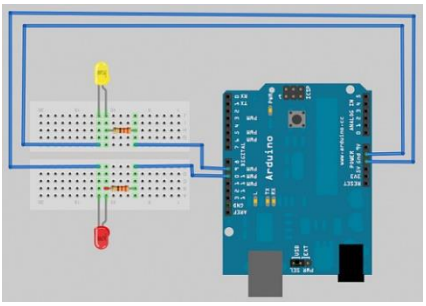
Berbeda dengan sensor DHT 11, sensor LDR hanya memiliki 2 pin yaitu pin untuk daya, dan pin untuk data, masing - masing pin perlu disambung dengan pin yang sesuai pada mikrokontroler wemos D1 R2, dimana pin VCC pada LDR dihubungkan dengan pin 5V pada mikrokontroller menggunakan kabel jumper dan pin data pada LDR dihubungkan dengan pin A0 yang merupakan pin input analog pada mikrokontroller, rancangan kofigurasi untuk mengintegerasikan sensor LDR dengan mikrokontroler wemos D1 R2 dapat dilihat pada **Gambar 4.3** berikut:



Gambar 4.3 Integerasi LDR Dengan Mikrokontroler Wemos D1 R2

4.2.1.3 Perancangan Integerasi Lampu LED dengan Mikrokontroler

Pada sistem ini menggunakan 2 buah LED yang yang nantinya akan melakukan aksi sesuai dengan pesan yang diterima, LED perlu diintegerasikan dengan mikrokontroler wemos D1 R2 sebagai sarana penerima pesan dan pengolah pesan yang nantinya menjadi acuan untuk LED melakukan aksi, rancangan kofigurasi untuk mengintegerasikan lampu LED dengan mikrokontroler wemos D1 R2 dapat dilihat pada **Gambar 4.4** berikut:

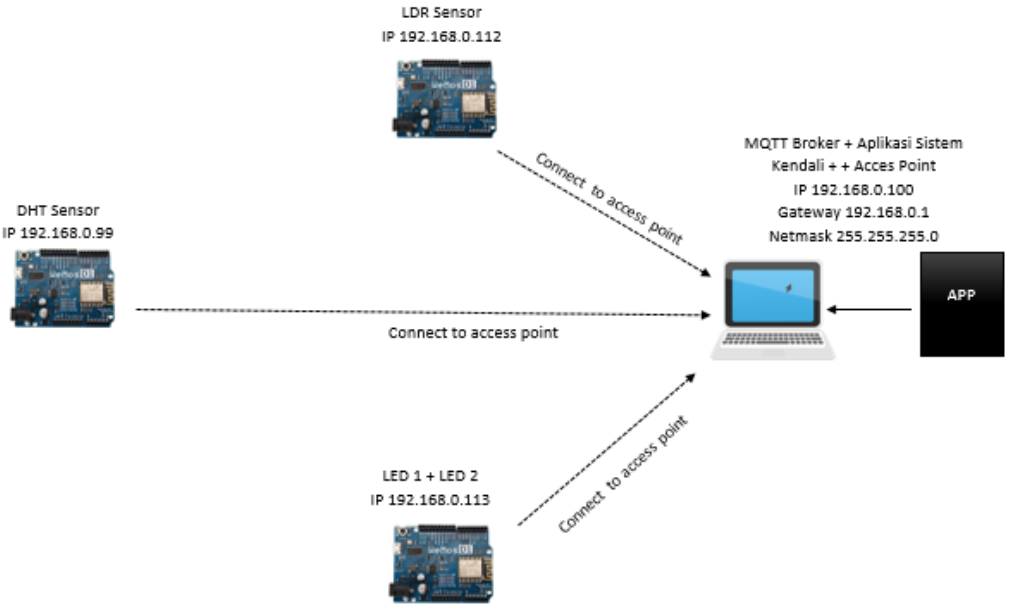


Gambar 4.4 Integerasi LED Dengan Mikrokontroler Wemos D1 R2

4.2.2 Perancangan Jaringan

4.2.2.1 Perancangan Arsitektur Jaringan

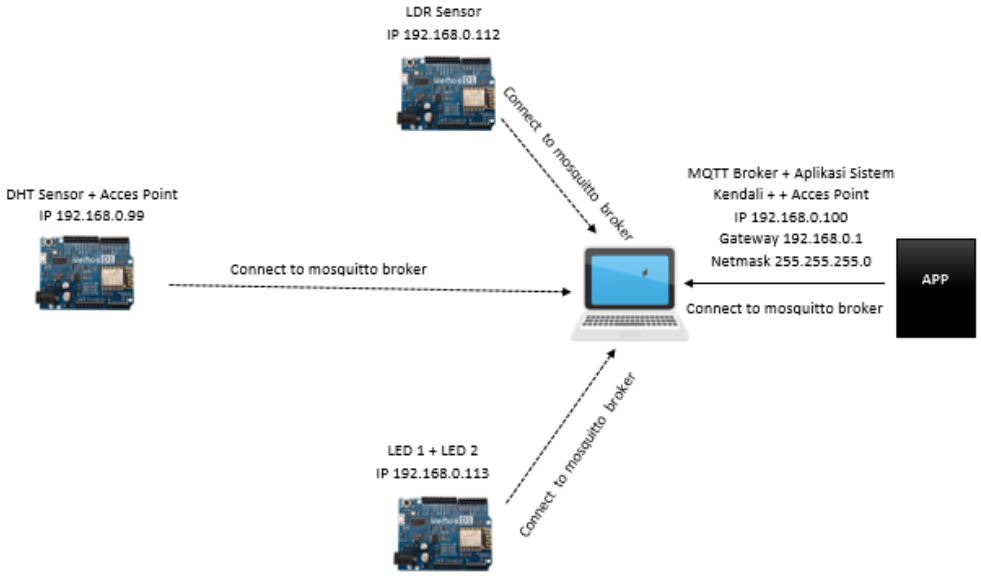
Arsitektur jaringan yang dibuat adalah yang berbentuk infrastruktur dimana nantinya komputer server akan berfungsi sebagai akses point, dan seluruh mikrokontroler akan terhubung pada akses point tersebut untuk membentuk sebuah sistem, pembuatan akses point pada komputer server memanfaatkan salah satu fitur dari sistem operasi windows yaitu *hosted network*. Perancangan arsitektur jaringan dapat dilihat pada **Gambar 4.5** berikut:



Gambar 4.5 Rancangan Arsitektur Jaringan

4.2.2.2 Perancangan Koneksi Sistem Dengan Mosquitto Broker

Setelah melakukan perancangan terkait dengan arsituktur jaringan agar sistem dapat saling terintegerasi dan terhubung di dalam satu jaringan, penggunaan protokol *MQTT* membutuhkan sebuah broker, pada penelitian ini digunakan mosquitto broker, broker yang nantinya akan menjadi perantara antara pengirim dan penerima pesan akan ditempatkan pada komputer server yang akan menjadi pusat kendali pada sistem kendali *smarthome* ini, nantinya semua mikrokontroller yang telah tergabung di dalam koneksi sistem akan terhubung dengan broker mosquitto yang berada pada komputer server, perancangan koneksi dengan mosquitto broker dapat dilihat pada **Gambar 4.6** berikut:



Gambar 4.6 Rancangan Koneksi Sistem Dengan Mosquitto Broker

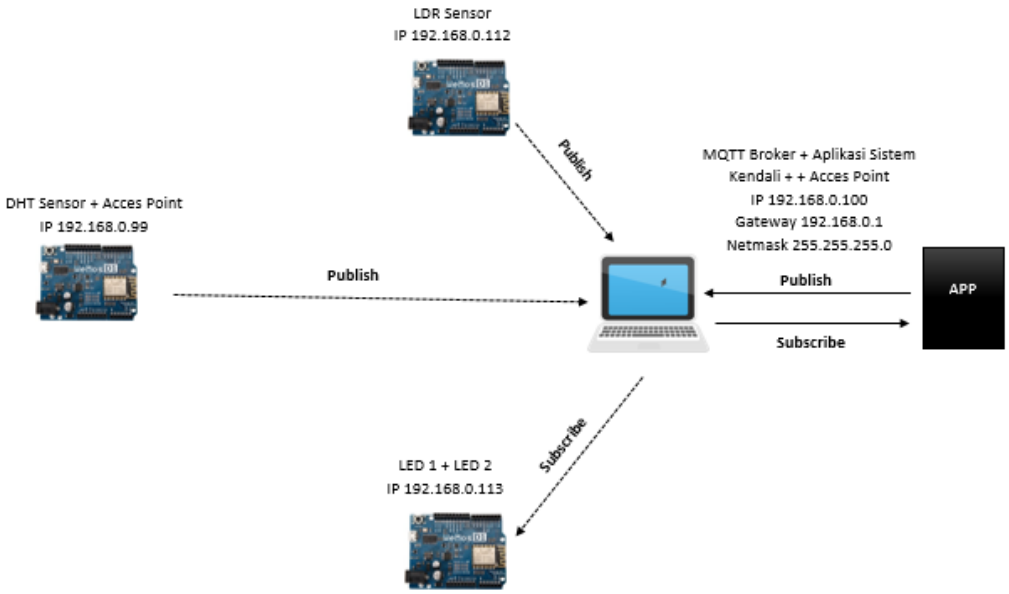
4.2.2.3 Perancangan *Publish/Subscribe* Sistem

Perancangan *publish/subscribe* adalah perancangan terkait dengan komunikasi berdasarkan protokol *MQTT* yang mempunyai tipe komunikasi *publish/subscribe*, perancangan ini akan menerangkan peranan dari masing - masing perangkat yang terdapat di dalam sistem dilihat dari fungsi *publish subscribe*nya, peranan dari masing – masing perangkat dapat dilihat pada **Tabel 4.1** berikut:

IP	Peran
192.168.0.99	*Publisher*
192.168.0.100	Broker + *Subscriber* + *Publisher*
192.168.0.112	*Publisher*
192.168.0.113	*Subscriber*

Tabel 4.1 Peran *Publish/Subscribe*

Di dalam sistem yang dikembangkan nantinya gambaran terkait dengan peranan dari masing masing perangkat yang terhubung didalam sistem dapat dilihat pada **Gambar 4.7** berikut:



Gambar 4.7 Rancangan Peranan *Publish/Subscribe*

4.2.3 Perancangan Perangkat Lunak

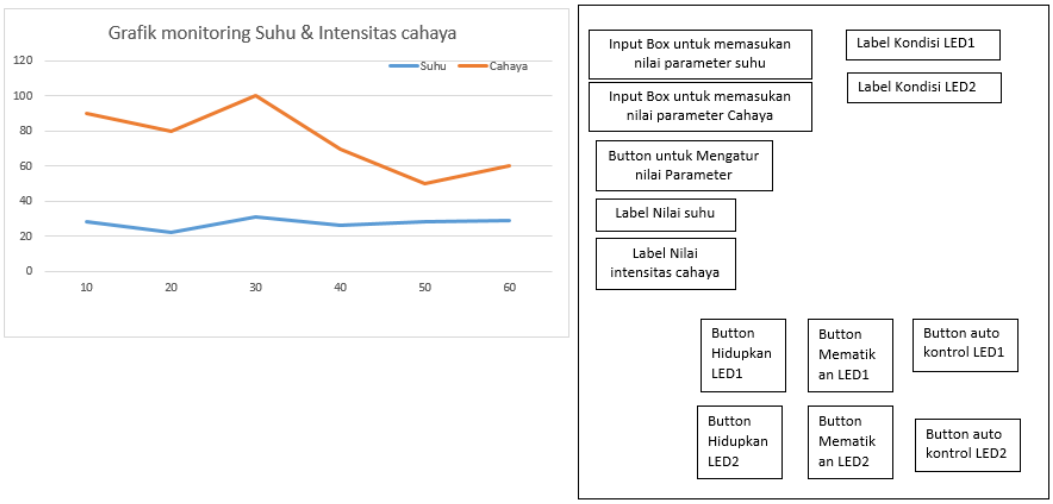
4.2.3.1 Perancangan Aplikasi Sistem Kendali

Di dalam sebuah sistem kendali diperlukan aplikasi sebagai perantara antara pengguna dengan siste, aplikasi yang dibuat pada penelitian ini harus dapat melakukan fungsi kontroling dan monitoring terhadap sistem, selain itu aplikasi juga harus bisa bekerja sebagai server dimana setiap data yang diterima akan disimpan ke dalam database redis. Aplikasi ini dituliskan dalam bahasa pemrograman *python* selain itu aplikasi web ini harus memliki fitur – fitur sebagai berikut:

1. Aplikasi harus dapat terhubung dengan Mosquitto broker.
2. Aplikasi harus dapat terhubung dengan database redis.
3. Aplikasi dapat menerima data sesuai dengan topik yang di*subscribe* yang telah ditentukan sebelumnya.
4. Aplikasi harus dapat menyimpan data yang telah diterima kedalam database redis.
5. Aplikasi harus mampu menampilkan grafik terkait data sensor yang telah didapat secara realtime.
6. Aplikasi harus dapat menampilkan kondisi/state dari masing – masing lampu LED.
7. Aplikasi harus dapat melakukan perubahan parameter pengendalian lampu LED.
8. Aplikasi harus dapat mengirimkan perintah untuk menyalakan lampu LED 1.

9. Aplikasi harus dapat mengirimkan perintah untuk menyalakan lampu LED 2.
10. Aplikasi harus dapat mengirimkan perintah untuk mematikan lampu LED 1.
11. Aplikasi harus dapat mengirimkan perintah untuk mematikan lampu LED 2.
12. Aplikasi harus dapat melakukan kontroling secara otomatis pada lampu LED 1 sesuai dengan parameter yang ditentukan.
13. Aplikasi harus dapat melakukan kontroling secara otomatis pada lampu LED 2 sesuai dengan parameter yang ditentukan.

Selain fungsi dibutuhkan juga user interface untuk memudahkan pengguna dalam melakukan monitroing dan kontroling terhadap sistem yang ada, rancangan user interface dapat dilihat pada **Gambar 4.8** berikut:



Gambar 4.8 Rancangan Aplikasi Sistem Kendali

4.2.3.2 Perancangan Database

Diperlukan sebuah database untuk menyimpan data yang telah didapat oleh sistem, data yang dimaksud sendiri adalah data hasi dari tangkapan sensor DHT 11 dan sensor LDR yang dikirimkan melalui mikrokontroler kepada aplikasi sistem kendali, pada sistem ini dibutuhkan 2 buah database redis yaitu 1 untuk menyimpan data suhu, dan 1 untuk menyimpan data intensitas cahaya, redis memliki beberapa tipe, dan yang digunakan untuk database pada sistem monitoring ini adalah tipe data list, dikarenakan struktur data list dapat menyimpan hasil dari nilai suhu dan intensitas cahaya secara berurutan dan memungkinkan memasukan nilai yang sama dalam waktu yang berbeda.Contoh Struktur database redis yang digunakan secara sederhana dapat terlihat pada **Tabel 4.2** dan **Tabel 4.3** berikut:

Tabel 4.2 Rancangan Database Suhu Redis

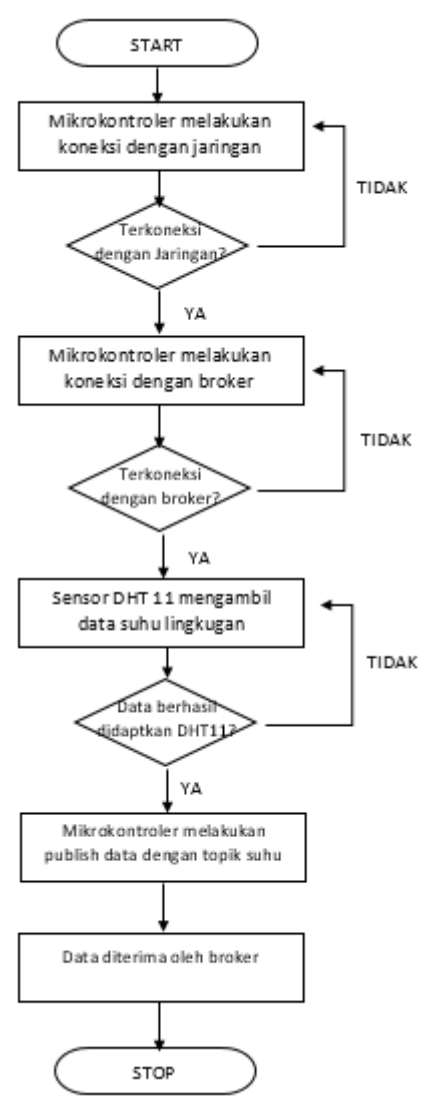
Nama List/Key	Value
Suhu	28.00
Suhu	29.00
Suhu	31.00

Tabel 4.3 Rancangan Database Intensitas Cahaya Redis

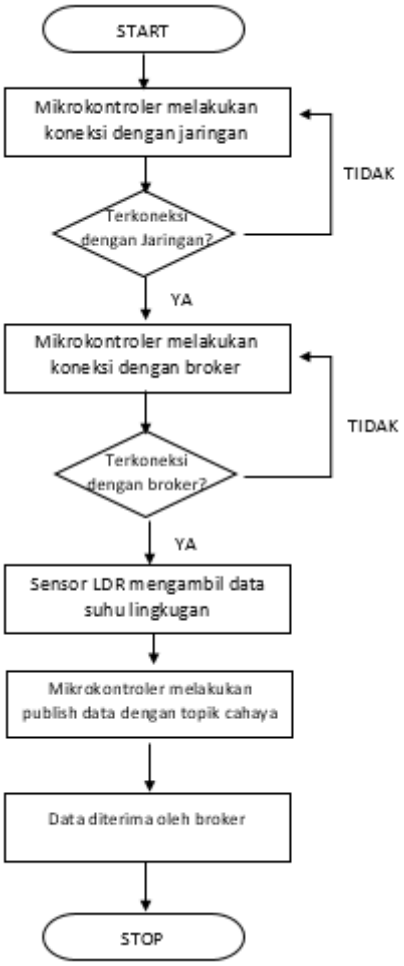
Nama List/Key	Value
Cahaya	100
Cahaya	50
Cahaya	25

4.2.3.3 Perancangan Monitoring Suhu dan Intensitas Cahaya dari Sensor

Perancangan monitoring digunakan untuk menggambarkan proses yang terjadi untuk melakukan monitoring dari data suhu dan intensitas cahaya yang diterima oleh sistem aplikasi server, perancangan monitroing meliputi proses komunikasi dan pertukaran antara mikrokontroler yang telah terintegrasi DHT11, mikrokontroler yang telah terintegerasi dengan LDR, broker, database, dan aplikasi sistem kendali, proses pertama dari sistem monitoring yang dibuat adalah pengambilan data oleh sensor lalu pengiriman data sensor menuju ke broker, alur proses pengambilan data sensor hingga pengiriman data hasil tangkapan sensor menuju ke broker dapat dilihat pada **Gambar 4.9** dan **Gambar 4.10** berikut:



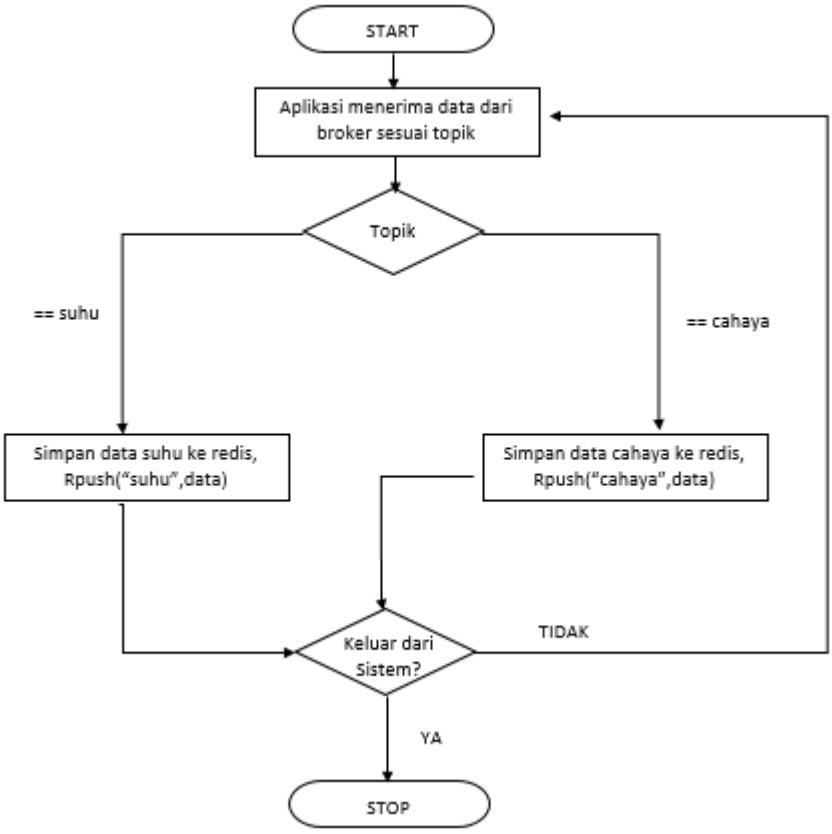
Gambar 4.9 Rancangan Pengiriman Data Suhu Ke Broker



Gambar 4.10 Rancangan Pengiriman Data Intensitas Cahaya ke Broker

Pengambilan data dan pengirman data pada broker seperti yang digambarkan pada diagram alur pada **Gambar 4.9** dan **Gambar 4.10** dilakukan oleh mikrokontroler yang telah terhubung oleh sistem, jika belum terkonkesi dengan jaringan mikrokontroler akan mencoba terhubung ke jaringan, setelah itu mikrokontroler akan mencoba terhubung dengan mosquitto broker, setelah terhubung dengan mosquitto broker, mikrokontroler baru akan melakukan pegembilan data melalui sensor kemudian mengirimkannya, komunikasi pengiriman pesan antara mikrokontroler dengan broker menggunakan protokol komunikasi *MQTT*, protokol *MQTT* mempunyai 2 buah fungsi yaitu fungsi *publish* dan fungsi *subscribe*, fungsi *publish* digunakan untuk mengirimkan pesan sesuai dengan topik kepada broker dan fungsi *subscribe* digunakan untuk menerima pesan sesuai dengan topik dari broker, setiap pengiriman data yang menggunakan protokol komunikasi *MQTT* maka data terlebih dahulu akan dikirimkan pada broker untuk selanjutnya oleh broker dikirimkan ke alamat tujuan. Setelah mikrokontroler mendapatkan data dari sensor, mikrokontroler memanggil method *publish*untuk mengirimkan pesan, mikrokontroler akan mem*publish* pesan dengan perintah *publish*(“topik”,data), itu berarti mikrokontroler yang teintegerasi dengan DHT 11 dan memperoleh nilai suhu akan mem*publish* pesan

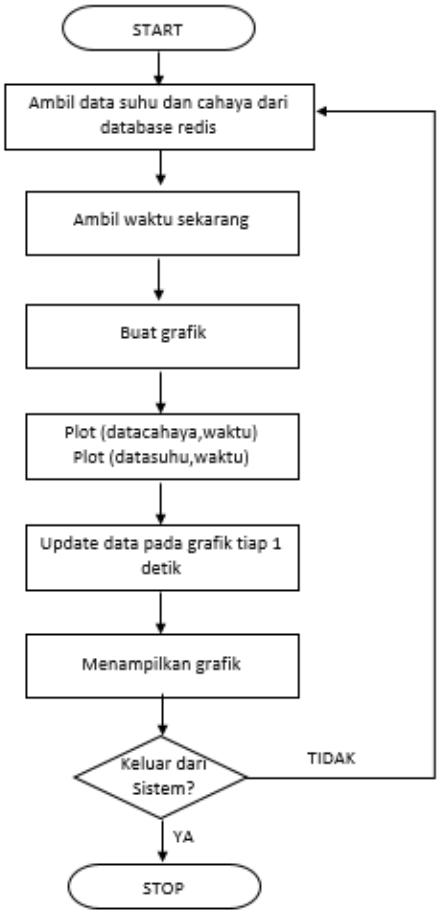
dengan contoh seperti *publish*(“suhu”,29.00) dimana “suhu” adalah topik dan 29.00 adalah nilai dengan tipe data float untuk nilai suhu yang telah didapatkan oleh sensor DHT11, dan mikrokontroler yang terintegerasi dengan sensor LDR akan mengirimkan data pada broker dengan mem*publish* pesan *publish*(“cahaya”,100) diamana cahaya adalah topik dan 100 adalah nilai intensitas cahaya yang didapat dari LDR, Setelah berhasil mendapatkan data sensor dan berhasil di*publish* pada mosquitto broker, selanjutnya broker akan mengirimkan data pada node yang melakukan *subscribe* pada topik suhu dan cayahaya, pada sistem ini node atau perangkat yang melakukan *subscribe* terhadap topik suhu dan topik cahaya adalah aplikasi sistem kendali, setelah menerima data aplikasi sistem kendali akan terlebih dahulu menyimpan data pada database redis sesuai dengan data yang diterimanya yaitu data suhu dan cahaya, alur penyimpanan data suhu dan intensitas cahaya pada database redis dapat dilihat pada **Gambar 4.11** berikut:



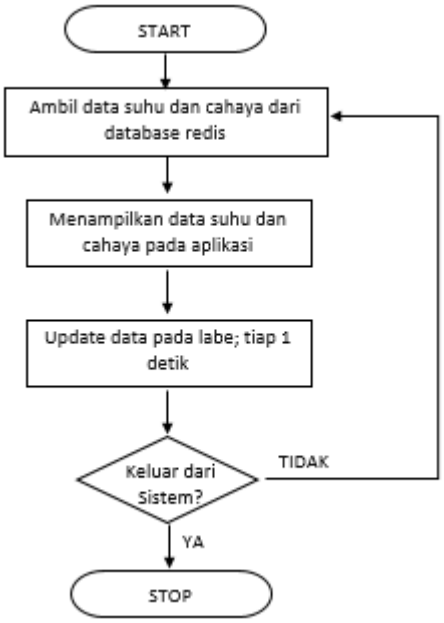
Gambar 4.11 Rancangan Penyimpanan Data Pada Database Redis

Penyimpanan data pada database redis dilakukan oleh aplikasi sistem kendali, proses yang dilakukan pada saat penyimpanan data ke database redis adalah, ketika aplikasi yang telah men*subscribe* topik suhu dan cahaya menerima data dengan topik suhu dan cahaya dari broker yang merupakan data yang dikirimkan oleh mikrokontroler, ketika data diterima aplikasi akan melakukan pilihan ketika pesan berisikan topik suhu maka aplikasi akan melakukan perintah rpush(“suhu”,29), dimana suhu adalah key pada database redis dan nilai 29 adalah

data value dari key tersebut, sama ketika aplikasi menerima data dengan topik cahaya, aplikasi akan melakukan perintah rpush(“cahaya”,100) diamana cahaya adalah key yang digunakan dan 100 adalah data value dari key tersebut. Setelah mendapatkan data dan menyimpannya didatabase redis, selanjutnya aplikasi harus dapat menampilkan grafik berdasarkan data suhu dan cahaya yang telah disimpan pada database redis sebelumnya, dan aplikasi juga harus menampilkan nilai data suhu dan cahaya pada aplikasi sebagai hasil akhir dari fungsi monitoring yang dimiliki oleh sistem, fungsi monitoring ini berguna untuk memberikan informasi kepada user terkait suhu dan intensitas cahaya dan perubahannya dari waktu ke waktu, proses menampilkan grafik pada aplikasi dimulai dari mengambil data suhu dana cahaya pada database redis, setelah itu mengambil nilai waktu saat ini dimana nilai waktu yang diambil adalah detik, setelah itu data dan nilai waktu dimasukan kedalam array dimana terdapat 2 buah array yaitu array X untuk nilai data, dan array Y untuk nilai waktu, untuk menampilkan data suhu dana chaaya dibutuhkan 4 buah array yaitu array X1 untuk nilai suhu, Y1 untuk nilai waktu dan X2 untuk nilai cahaya, dan Y2 untuk nilai waktu, kemudian ke 4 array tersebut di *plot* pada grafik yang telah dibuat plot(X1,Y2) untuk grafik nilai suhu perdetiknya, dan plot(X2,Y2) untuk grafik nilai cahaya perdetiknya kemudian array akan dihapus setiap 1menit sekali sehingga grafik yang ditampilkan merupakan grafik pada menit tersebut, dan untuk menampilkan nilai suhu dan cahaya pada label di aplikasi dengan cara mengambil data dari database redis kemudian menampilkannya pada label setelah itu mengupdate nilai pada label setiap 1 detik. Rancangan menampilkan grafik dan data nilai pada aplikasi dapat dilihat pada **Gambar 4.12** dan **Gambar 4.13** berikut:



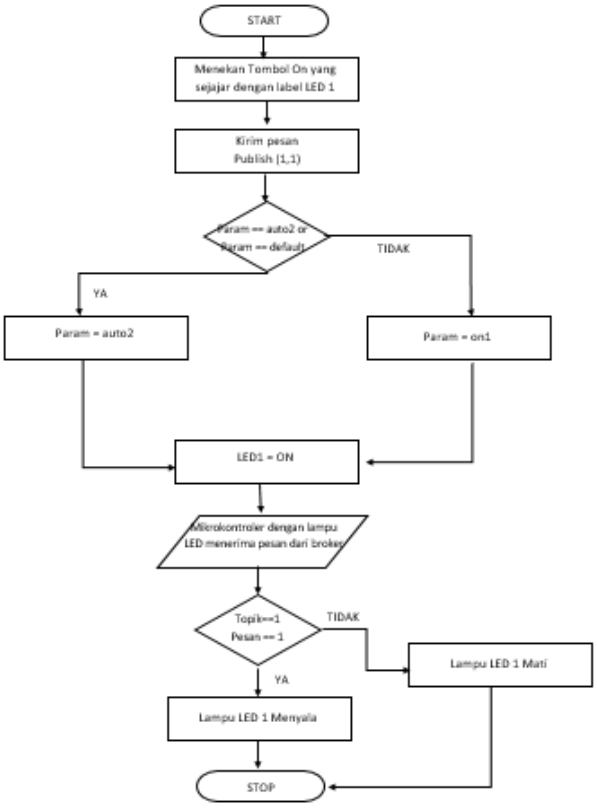
Gambar 4.12 Rancangan Menampilkan Grafik Data Suhu Dan Cahaya



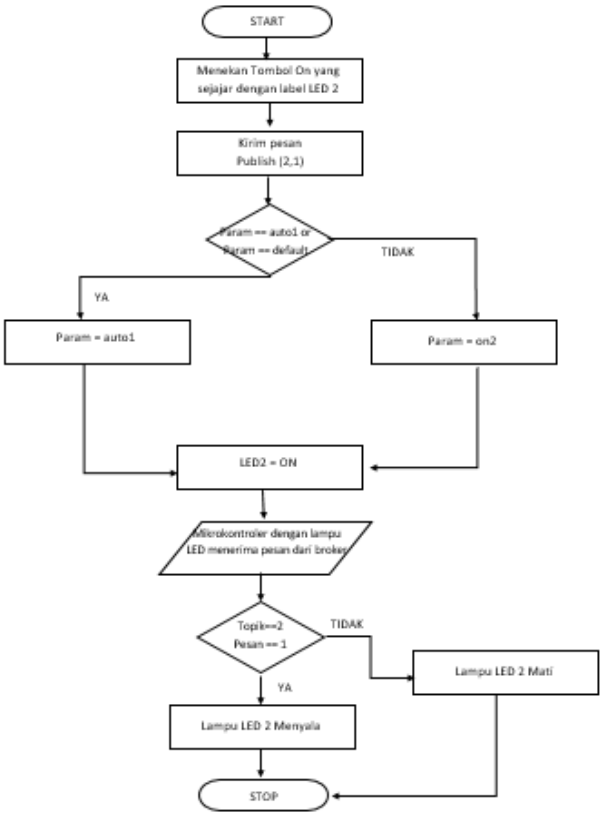
Gambar 4.13 Rancangan Menampilkan Data Suhu dan Cahaya pada Label di Aplikasi

4.2.3.4 Perancangan Kontroling Lampu LED

Perancangan kontroling digunakan untuk merancang sistem kontroling terhadap lampu LED, sistem nantinya harus dapat menyalakan, dan mematikan lampu LED yang terpasang pada mikrokontroler, sistem juga harus dapat membuat lampu LED menyala dan mati secara otomatis berdasarkan nilai parameter suhu, dan nilai parameter cahaya, aplikasi juga harus dapat pengguna mengatur nilai dari parameter suhu dan parameter cahaya tersebut. Untuk dapat membuat sistem kontroling lampu LED dibutuhkan beberapa perancangan sebelumnya yaitu yang meliputi, mengirimkan pesan untuk menyalakan lampu LED, mengirimkan pesan untuk mematikan lampu LED, membuat sistem mengatur otomatis state dari lampu LED. Untuk alur perancangan menyalakan lampu LED dapat dilihat pada **Gambar 4.14** dan **Gambar 4.15** berikut:

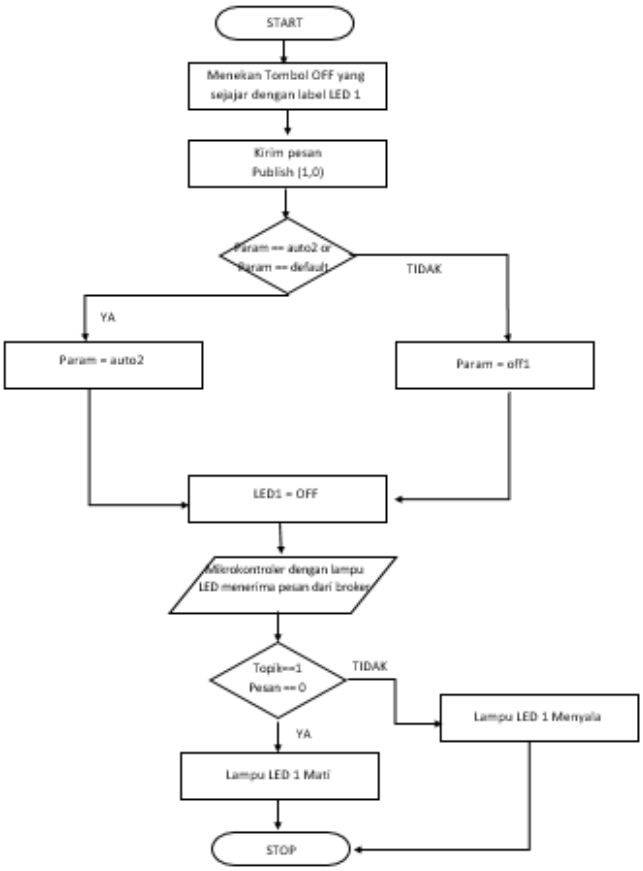


Gambar 4.14 Rancangan Menyalakan Lampu LED 1

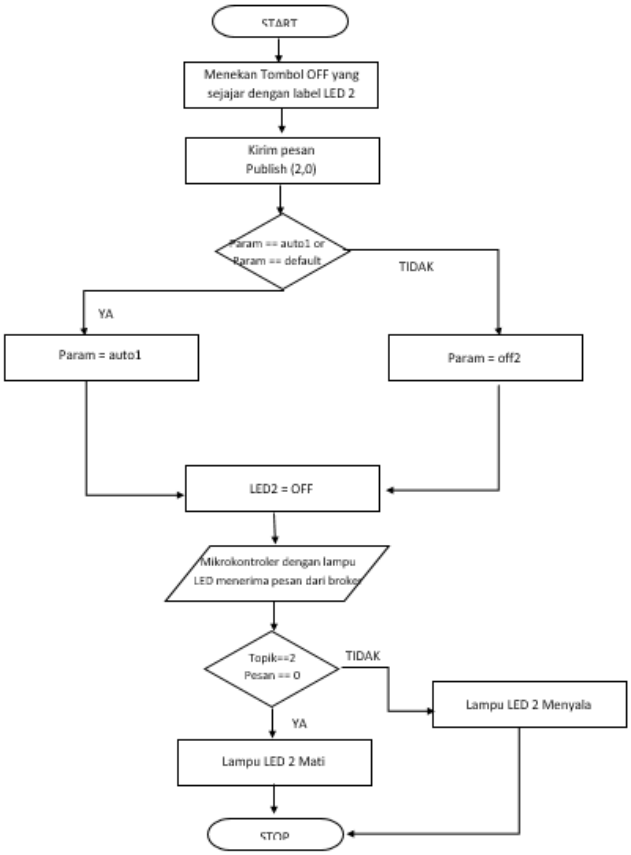


Gambar 4.15 Rancangan Menyalakan Lampu LED 2

Untuk menyalakan lampu LED 1 mau lampu LED 2, aplikasi akan mem*publish* pesan pada broker isi pesan yang di *publish* oleh aplikasi bergantung dengan tombol mana yang ditekan oleh pengguna, terdapa 2 buah tombol ON yang masing – masing sejajar dengan label LED tertentu, untuk tombol ON yang sejajar dengan label LED1 maka akan membuat aplikasi mem*publish* pesan dengan topik = 1 dan pesan = 1, yang akan menjadi input bagi mikrokontroler yang terintegerasi dengan lampu LED, dimana topik menentukan id dari lampu LED, 1 untuk LED1 dan 2 untuk LED2, dan pesan menentukan state dari lampu LED dimana 1 untuk state menyala dan 0 untuk mati, selain itu terdapat sebuah parameter bernama param yang akan menjadi pengatur method callback dari *MQTT*, variabel ini nantinya akan menyeleksi agar sitem aplikasi tetap mendapatkan data suhu dan cahaya dari mikrokontroler tetapi tidak mempengaruhi kondisi atau state dari lampu LED yang ada, hal ini dikarenakan karena tujuan dari tombol ON adalah membuat lampu LED tetap menyala tanpa mempertimbangkan nilai dari data hasil tangkapan sensor. Sama dengan perancangan dalam menyalakan lampu LED, perancangan untuk mematikan lampu LED hanya membedakan isi pesan yang di *publish* dimana aplikasi akan mem*publish* pesan dengan nilai 0 yang akan menjadi input untuk mikrokontroler yang terintegerasi dengan lampu LED untuk mematikan lampu LED tersebut, perancangan untuk mematikan lampu LED dapat dilihat pada **Gambar 4.16** dan **Gambar 4.17** berikut:

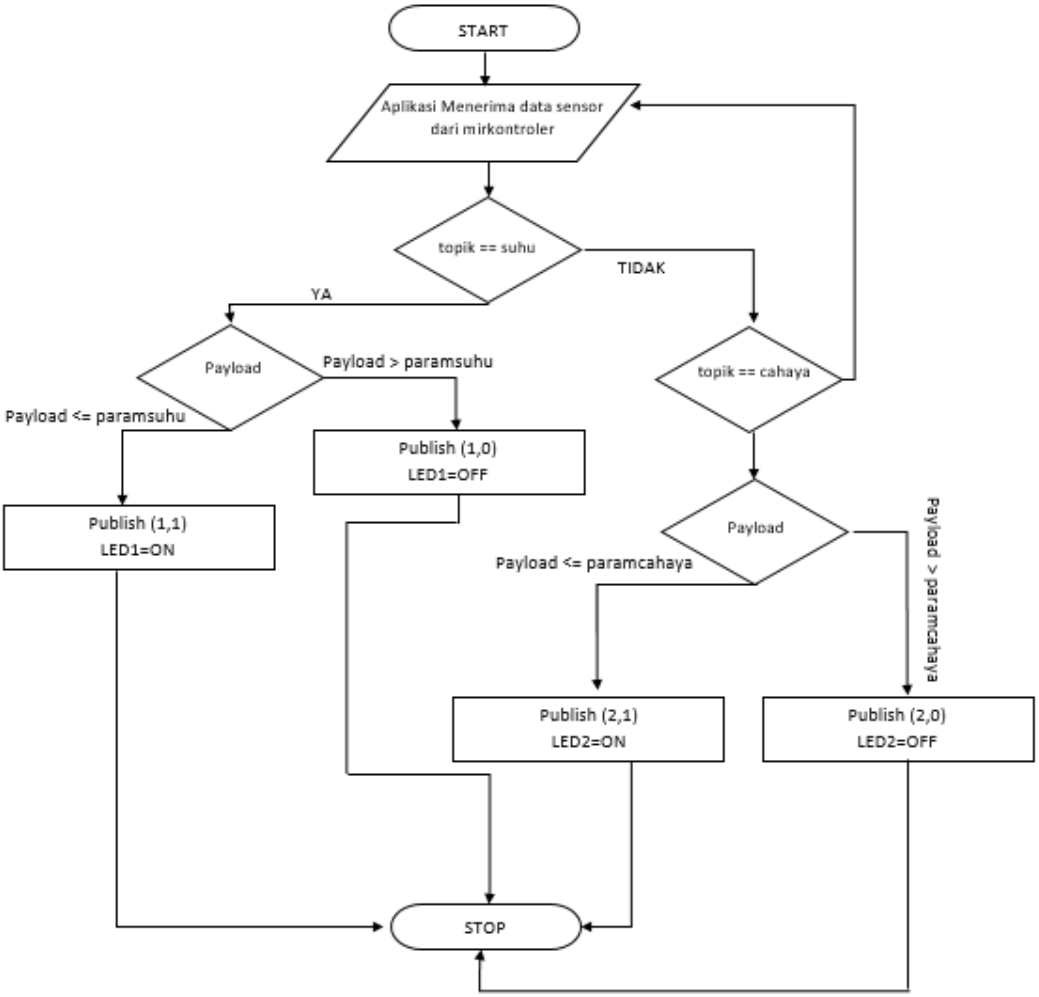


Gambar 4.16 Rancangan Mematikan Lampu LED 1

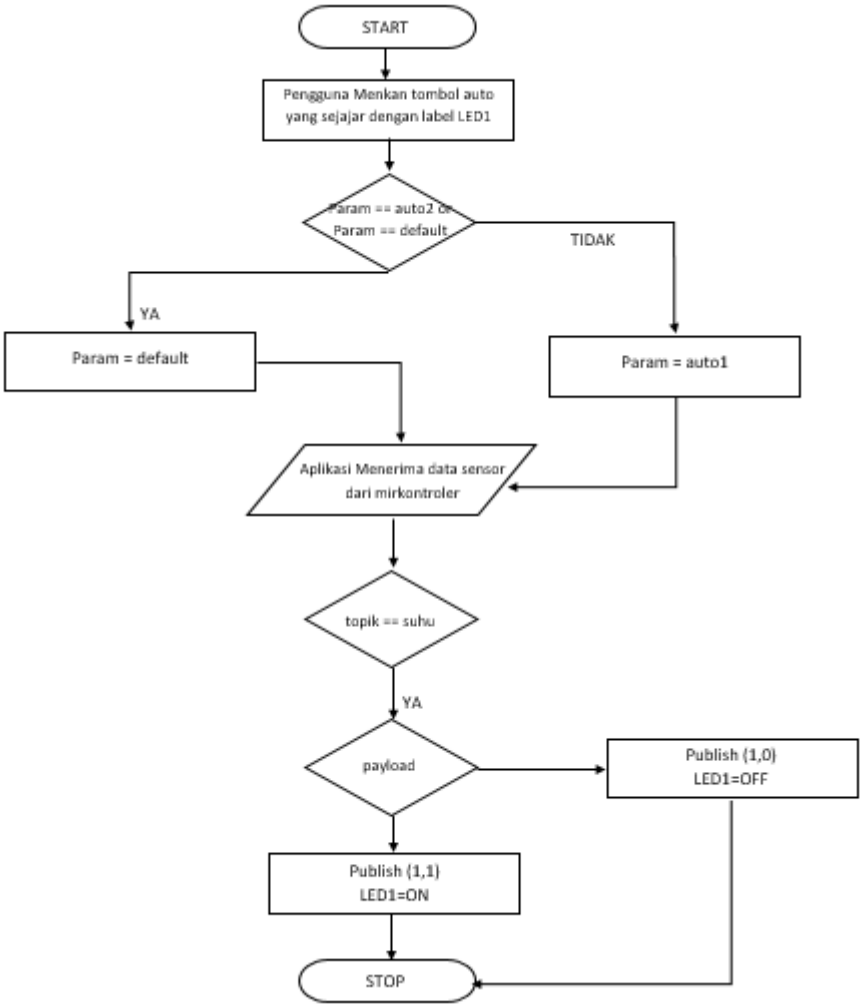


Gambar 4.17 Rancangan Mematikan Lampu LED 2

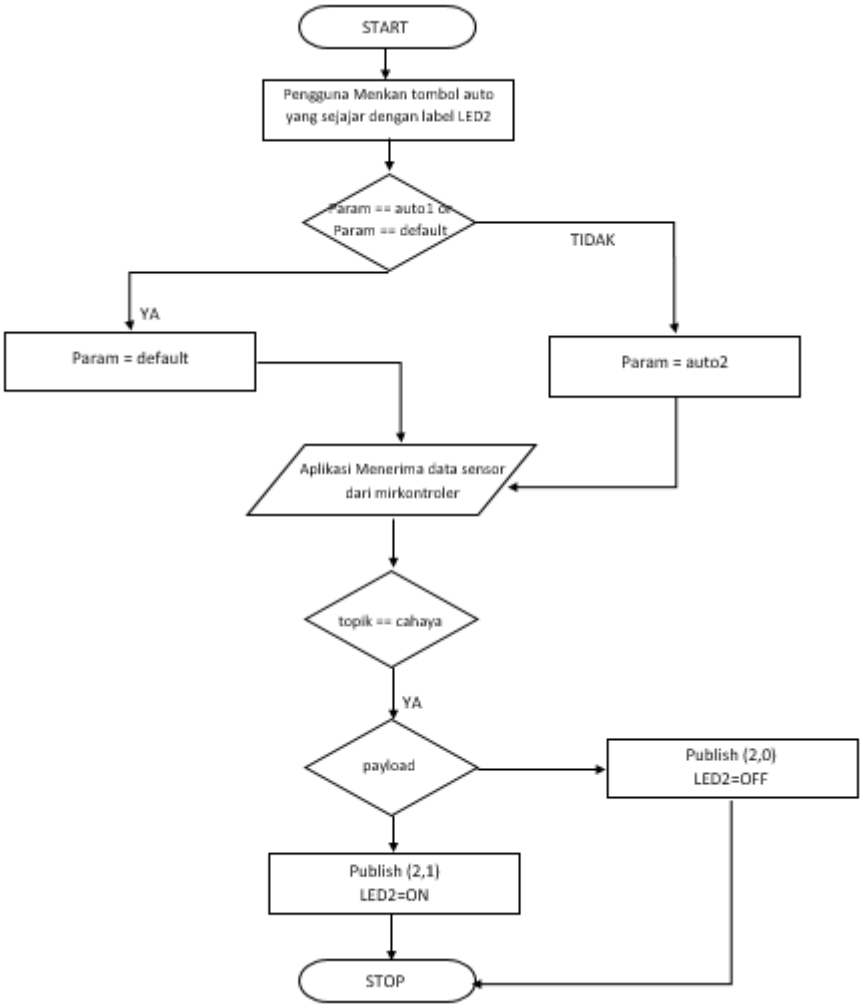
Setelah dapat mematikan lampu LED secara manual dengan menekan tombol OFF pada aplikasi, aplikasi harus dapat lampu LED pada mikrokontroler menyala dan mati sesuai dengan nilai parameter suhu atau cahaya yang telah ditentukan, sistem nantinya akan membandingkan nilai dari yang diperoleh sensor dengan parameter yang telah ditentukan, ketika nilai dari suhu berada lebih kecil dari nilai parameter yang ditentunkan maka lampu LED akan menyala, lampu LED1 akan menyala atau mati berdasarkan nilai sensor suhu yang dibandingkan dengan nilai parameter suhu, dan lampu LED2 akan menyala atau mati berdasarkan nilai sensor cahaya yang dibandingkan dengan nilai parameter cahaya, fitur ini merupakan fitur default ketika apalikasi dijalankan dan nilai paramter awal untuk suhu adalah 30 dan nilai awal parameter awal untuk cahaya adalah 50, fitur ini juga dapat terhenti dengan cara menekan tombol ON atau OFF untuk menyalakan atau mematikan lampu LED secara manual, dan fitur ini dapat dinaktifkan kembali dengan menekan tombol auto yang berada sejajar dengan label LED 1 untuk menyalakan dan mematikan otomatis lampu LED 1 dan menekan tombol auto yang berada sejajar dengan label LED 2 untuk menyalakan dan mematikan otomatis lampu LED 2, perancangan untuk menyalakan dan mematikan secara otomatis lampu LED 1 dan lampu LED 2 dapat dilihat pada **Gambar 4.18**, **Gambar 4.19**, dan **Gambar 4.20** berikut:



Gambar 4.18 Rancangan Default Menyalakan dan Mematikan Lampu LED Secara Otomatis

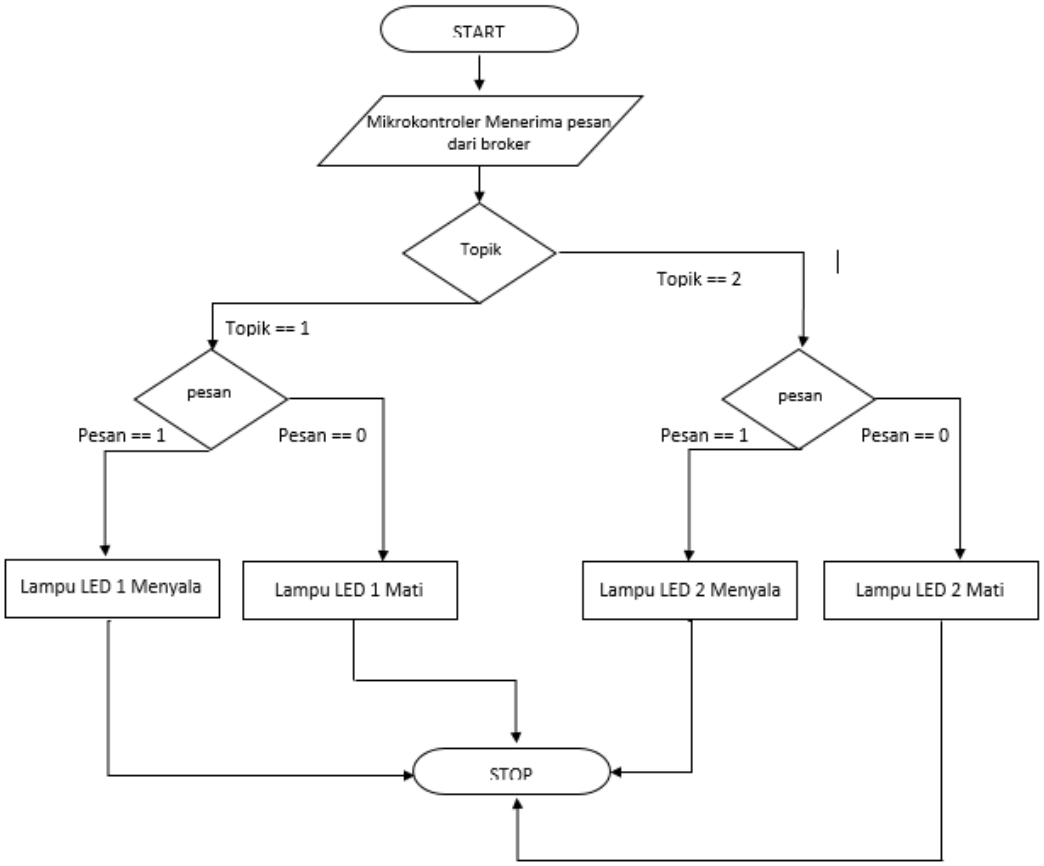


Gambar 4.19 Rancangan Menyalakan dan Mematikan Lampu LED Secara Otomatis Dengan Menekan Tombol Auto yang Sejajar dengan Label LED 1



Gambar 4.20 Rancangan Menyalakan dan Mematikan Lampu LED Secara Otomatis Dengan Menekan Tombol Auto yang Sejajar dengan Label LED 2

Untuk menyalakan lampu LED pada tahap akhir kontroling diperlukan perintah pada mikrokontroler yang terintegerasi dengan lampu LED untuk dapat menyeleksi pesan yang nantinya diterima untuk menyalakan ataupun mematikan lampu LED yang terintegerasi dengannya perancangan menyalakan dan mematikan lampu LED pada mikrokontroler dapat dilihat pada **Gambar 4.21** berikut:



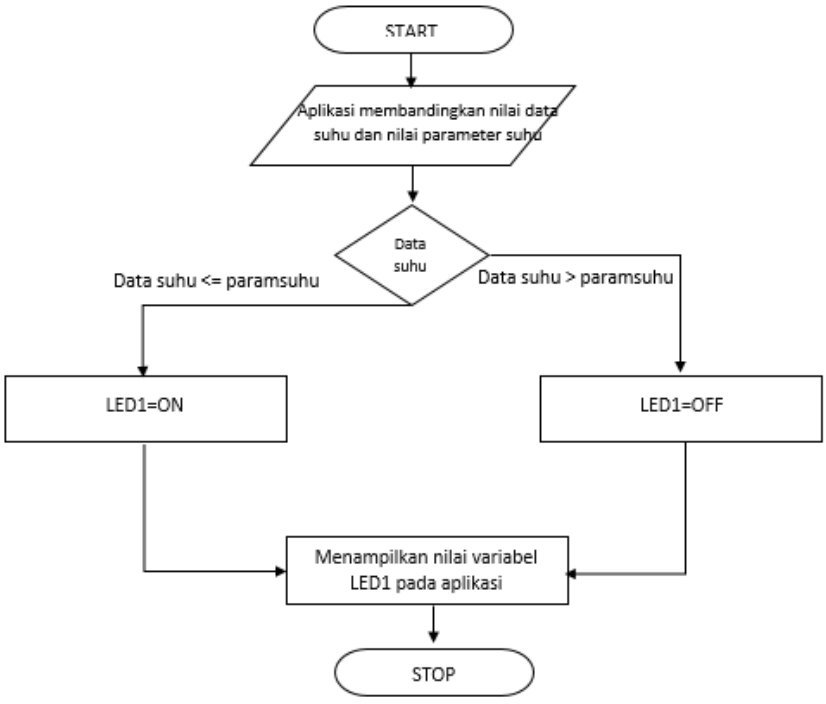
Gambar 4.21 Perancangan Menyalakan dan Mematikan Lampu LED pada Mikrokontroler

Setelah perancangan untuk menyalakan dan mematikan lampu LED secara otomatis diperlukan juga fitur untuk mengatur nilai parameter suhu dan parameter cahaya, agar pengguna dapat dengan leluasa menentukan parameter sesuai dengan kondisi yang terjadi, pengaturan terhadap parameter suhu dan cahaya ini nantinya akan menggunakan input langsung dari pengguna, dimana penggunakan akan memasukan nilai pada box input untuk paramter cahaya atau parameter suhu, kemudian menekan tombol Enter yang akan mengeksekusi perintah perubahan parameter suhu dan parameter cahaya sesuai denga inputan yang di input oleh pengguna, perancangan untuk mengatur nilai parameter suhu dan nilai parameter cahaya dapat dilihat pada **Gambar 4.22** berikut:

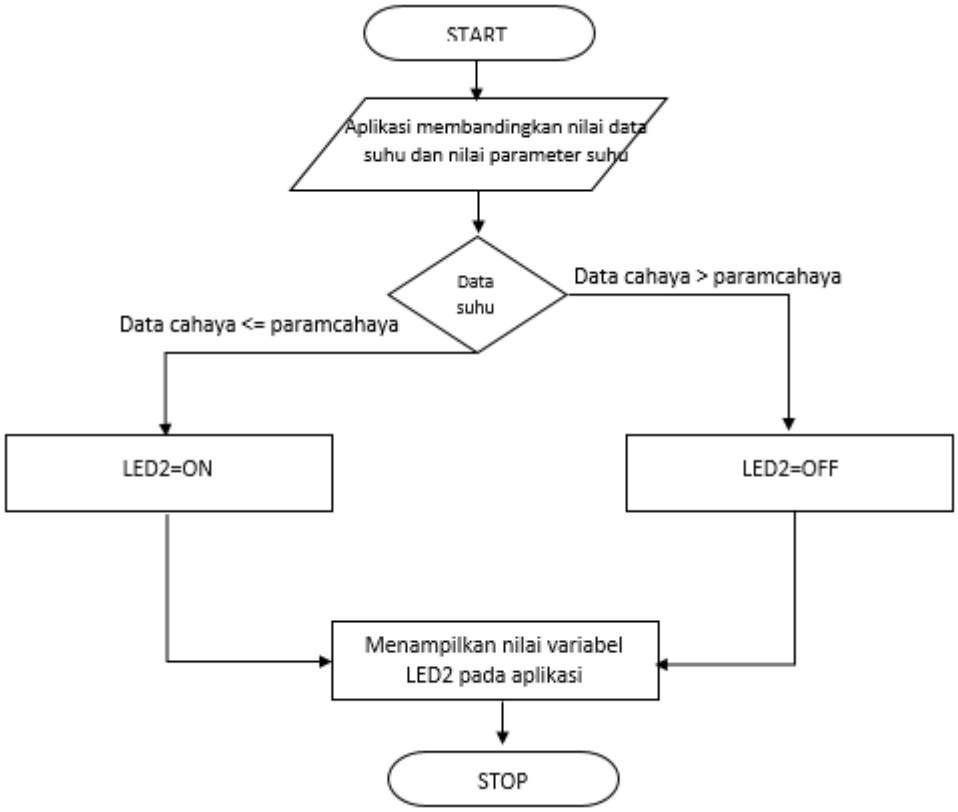
Gambar 4.22 Rancangan Input Nilai Parameter Suhu dan Paramater Cahaya

4.2.3.5 Perancangan Monitoring Lampu LED

Setelah melakukan perancangan kontroling lampu LED dibutuhkan sebuah perancangan untuk mengetahui kondisi atau state dari LED 1 dan LED 2, hal ini diperlukan agar pengguna dapat mengetahui kondisi atau state dari LED 1 dan LED 2 melalu aplikasi sistem kendali, alur perancangan monitoring kondisi lampu LED dapat dilihat pada **Gambar 4.23 Gambar 4.24** berikut:



Gambar 4.23 Perancangan Monitoring Kondisi Lampu LED 1



Gambar 4.24 Perancangan Monitoring Kondisi Lampu LED 2

Kondisi lampu LED 1 dan lampu LED 2 dimonitoring dengan cara menampilkannya pada label di aplikasi sistem kendali, ketika lampu LED 1 dalam kondisi menyala maka aplikasi akan merubah nilai dari variabel LED1 menjadi ON ketika lampu LED 1 dalam kondisi mati aplikasi akan merubah nilai variabel LED1 menjadi OFF, variabel LED1 merupakan variabel yang menyimpan kondisi LED 1 setelah itu aplikasi akan menampilkan nilai dari variabel LED1 pada label diaplikasi sistem kendali, sama seperti lampu LED 1 ketika lampu LED 2 dalam kondisi menyala maka aplikasi akan merubah nilai dari variabel LED2 menjadi ON ketika lampu LED 2 dalam kondisi mati aplikasi akan merubah nilai variabel LED2 menjadi OFF, variabel LED1 merupakan variabel yang menyimpan kondisi LED 2 setelah itu aplikasi akan menampilkan nilai dair variabel LED2 pada label diaplikasi sistem kendali.

BAB 5 IMPLEMENTASI

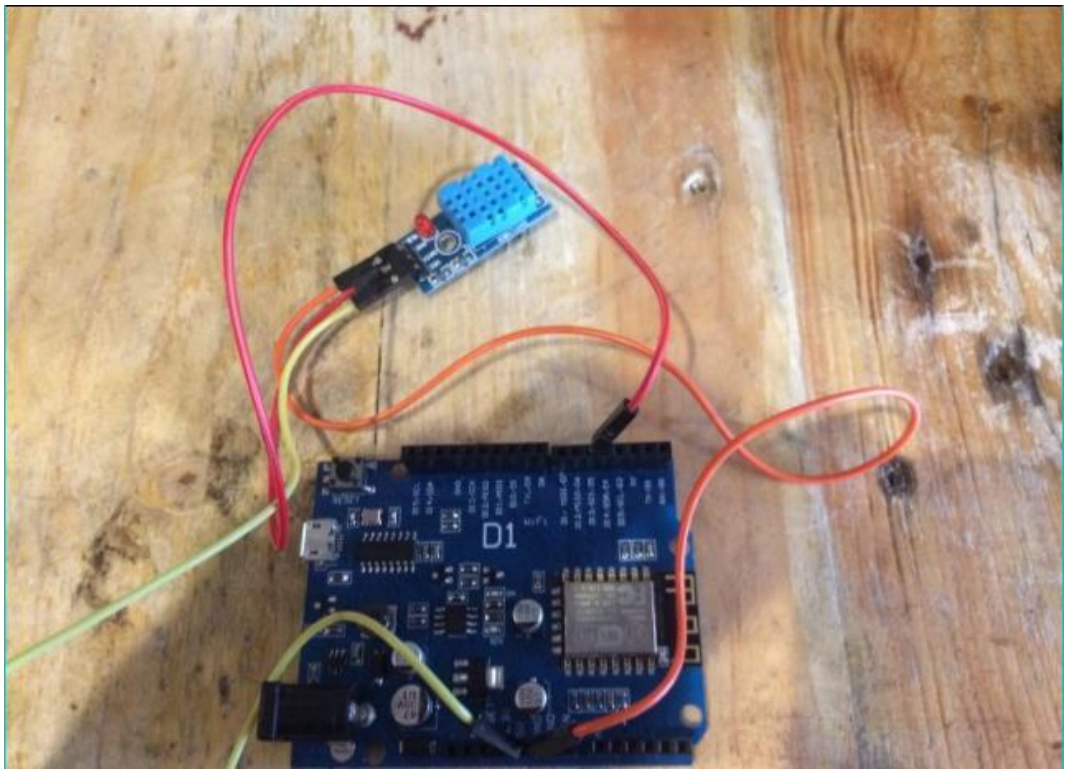
Implementasi adalah bagian terpenting dalam sebuah pengembangan sistem berhasil atau tidaknya sistem yang dibuat ditentukan oleh keberhasilan dari implementasinya, implementasi juga bertujuan untuk memenuhi tujuan dari penelitian yang dilakukan, implementasi dilakukan berdasarkan dengan perancangan yang telah dibuat pada bab sebelumnya.

5.1 Implementasi Perangkat Keras

Implementasi perangkat keras berisi tentang penjelasan bagaimana melakukan implementasi pengintegrasian antara mikrokontroler dengan perangkat perangkat yang terhubung dengannya, yaitu implementasi penintegrasian sensor DHT 11 dengan mikrokontroler wemos D1 R2, implementasi pengintegrasian sensor LDR dengan mikrokontroler wemos D1 R2, dan pengintegrasian lampu LED dengan mikrokontroler wemos D1 R2.

5.1.1 Implementasi Integrasi DHT 11 dengan Mikrokontroler

Seperti yang telah dijelaskan pada perancangan sebelumnya untuk mengintegrasikan DHT 11 dengan mikrokontroler wemos D1 R2 diperlukan untuk menghubungkan pin DHT11 dengan pin Wemos D1 R2, hasil integrasi DHT11 dengan mikrokontroler wemos D1 R2 dapat dilihat pada **Gambar 5.1** berikut:

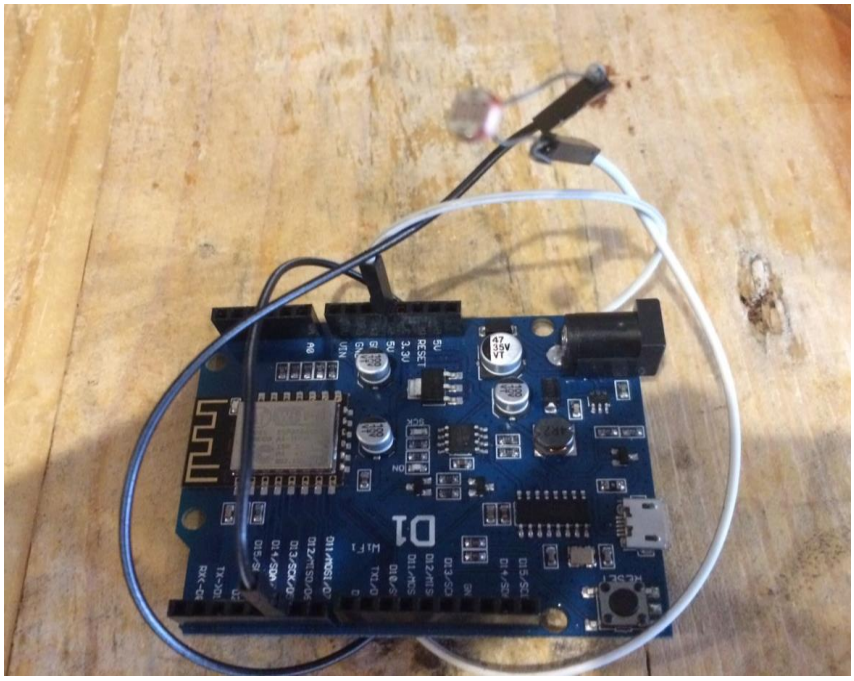


Gambar 5.1 Implementasi Integrasi DHT11 dengan Mikrokontroler

Pada gambar **Gambar 5.1** di atas untuk menghubungkan pin DHT 11 dengan pin mikrokontroler Wemos D1 R2 menggunakan kable jumper sebagai alat bantu. Kabel berwarna oranye menghubungkan pin GND pada DHT 11 dengan pin GND pada mikrokontroler wemos D1 R2, kabel berwarna merah menghubungkan pin Data pada DHT 11 dengan pin D13/D5 pada mikrokontroler wemos D1 R2 dan kabel berwarna kuning menghubungkan pin VCC pada DHT 11 dengan pin 5V pada mikrokontroler wemos D1 R2 sebagai sumber daya untuk sensor DHT 11.

5.1.2 Implementasi Integrasi Sensor LDR dengan Mikrokontroler

Sama seperti melakukan integrasi DHT 11 dengan mikrokontroler, implementasi integrasi antara sensor LDR dengan mikrokontroler wemos D1 R2 juga dilakukan dengan cara menghubungkan pin pada sensor LDR dengan pin pada mikrokontroler wemos D1 R2 dengan bantuan kabel jumper, hasil integrasi sensor LDR dengan mikriokontroler wemos D1 R2 dapat dilihat pada **Gambar 5.2** berikut:



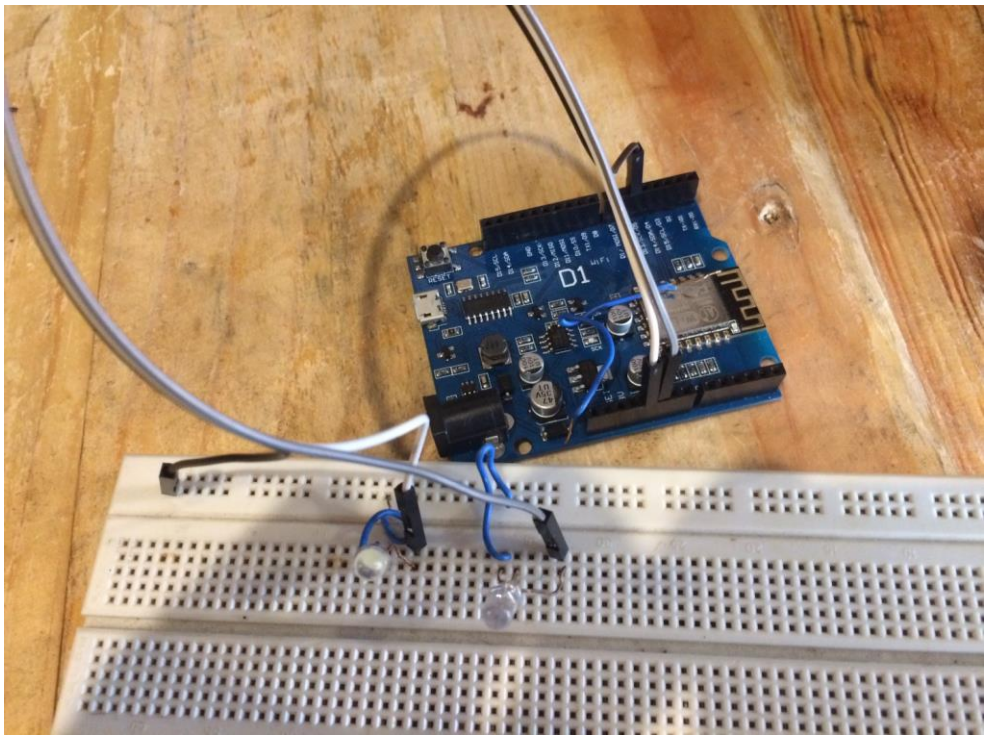
Gambar 5.2 Implementasi Integrasi Sensor LDR dengan Mikrokontroler

Pada **Gambar 5.2** kabel jumper berwarna putih menghubungkan antara pin sensor LDR dengan pin 5V pada mikrokontroler sebagai sumber daya untuk sensor dan kabel jumper berwarna hitam menghubukan pin sensor LDR dengan pin A0 pada mikrokontroler wemos D1 R2 sebagai pin yang akan menerima input data dari hasil tangkapan data sensor LDR.

5.1.3 Implementasi Integrasi Lampu LED dengan Mikrokontroler

Pada Implementasi Integrasi lampu LED dengan mikrokontroler wemos D1 R2 menggunaka 2 buah lampu LED dengan 2 warna berbeda yaitu warna biru dan putih, penintegrasian lampu LED menggunakan bantuan *board* untuk menempatkan lampu LED dan juga menggunakan bantuan kaberl jumper untuk menghubungkan lampu LED dengan mikrokontroler wemos D1 R2, implementasi

pengintegrasian lampu LED dengan mikrokontroler wemos D1 R2 dapat dilihat pada **Gambar 5.3** berikut:



Gambar 5.3 Implementasi Integrasi Lampu LED dengan Mikrokontroler

Pada **Gambar 5.3** 2 buah lampu LED dipasangkan pada *board* kemudian kabel jumper berwarna hitam dihubungkan pada board sebagai sumber daya untuk lampu LED. Kabel berwarna putih menghubungkan lampu LED 1 dengan PIN D13/D5 pada mikrokontroler wemos D1 R2, dan kabel jumper berwarna abu – abu menghubungkan lampu LED dengan pin D15/D7 pada mikrokontroler wemos D1 R2 sebagai input untuk lampu LED melakukan aksi.

5.2 Implementasi Jaringan

5.2.1 Implementasi Arsitektur Jaringan

Pada implementasi arsitektur jaringan, komputer server akan berfungsi sebagai akses point dimana dalam pembentukan akses point komputer server menggunakan fitur *hosted network* dari sistem operasi window. Untuk langkah - langkah pengimplementasian akses point pada komputer server adalah sebagai berikut:

1. Masuk ke dalam *command prompt (administrator)*
2. Setelah itu masukan perintah *NETSH WLAN set hostednetwork mode=allow ssid=hudan key=12345678.* Dimana hudan adalah nama ssid akses point yang di buat dan 12345678 adalah password yang di gunakan untuk terhubung dengan akses point.

3. Kemudian jalankan *hosted network* yang telah di buat dengan perintah *NETSH WLAN start hostednetwork*

Setelah berhasil membuat akses point berikutnya adalah menghubungkan perangkat lainnya pada akses point, untuk pengimplementasian menghubungkan mikrokontroler lainnya pada akses point dapat dilihat pada **Kode Program 5.1** berikut:

```
1  WiFi.begin(ssid, password);
2  WiFi.config(ip,gateway,subnet);
3  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
4      Serial.println("Connection Failed! Rebooting...");
5      delay(5000);
6      Serial.print(".");
7      ESP.restart();
8    }
```

Kode Program 5.1 Implemetasi Koneksi Mikrokontroler pada Akses Point

Pada **Kode Program 5.1** adalah kode program yang digunakan sebagai perintah untuk mikrokontroler yang terintegrasi dengan sensor LDR dan mikrokontoler yang terintegrasi dengan lampu LED untuk terhubung dengan akeses point penggalan kode tersebut ditempatkan pada method setup() untuk dijalankan ketika mikrokontroler menyala, penjelasan dari **Kode Program 5.1** adalah sebagai berikut:

1. Baris 1 digunakan untuk terhubung dengan akeses point, parameter ssid dan password disesuaikan dengan ssid dan pasword yang digunakan oleh akses point.
2. Baris ke 2 digunakan untuk mengatur alamat mikrokontroler secara statis dengan parameter alamat ip, gateway, dan subnet masknya.
3. Baris ke 3-8 adalah perulangan while yang dilakukan ketika mikrokontroler gagal terkoneksi dengan akses point maka mikrokontroler akan mencoba kembali terhubung dengan akses point secara otomatis dengan jeda 5 detik. Sedangkan untuk menghubungkan komputer server dengan akses point menggunakan cara seperti pada umumnya ketika komputer ingin terhubung dengan jaringan wireless yaitu dengan memiliih jaringan yang ada.

5.2.2 Implementasi Koneksi Sistem dengan Mosquitto Brokker

Setelah semua perangkat terhubung dengan akses point yang sama, untuk menggunakan protokol komunikasi *MQTT* hal pertama yang perlu dilakukan adalah terhubung dengan broker, semua perangkat yang digunakan perlu terhubung dengan broker, karena broker akan menjadi perantara dalam proses pengirimian pesan, untuk implementasi menghubungkan mikrokontroler dengan broker dapat dilihat pada **Kode Program 5.2** berikut.

```
1   const char* MQTT_server = "192.168.0.100";
2   client.setServer(MQTT_server, 1883);
3   while (!client.connected()) {
4       Serial.print("Attempting MQTT connection...");
5       if (client.connect("LDR")) {
6         Serial.println("connected");
7       } else {
8         Serial.print("failed, rc=");
9         Serial.print(client.state());
10        Serial.println(" try again in 5 seconds");
11        delay(5000);
12      }
13    }
```

Kode Program 5.2 Implementasi Koneksi Mikrokontroler dengan Broker

Penjelasan **Kode Program 5.2** adalah sebagai berikut:

1. Baris 1 adalah inisialisasi variabel yang berisikan alamat ip broker.
2. Baris 2 untuk melakukan koneksi dengan broker dengan parameter alamat ip broker dan port yang digunakan.
3. Baris 3-13 digunakan untuk melakukan pengecekan koneksi dengan broker, ketika koneksi gagal dilakukan mikrokontroler akan mencoba terkoneksi kembali setiap 5 detik, setiap perangkat yang terhubung dengan broker perlu memiliki sebuah id yang unik dimana id ini di inisialisasi ketika perangkat terhubung dengan broker, sepert yang terlihat pada baris ke 5.

Sementara untuk implementasi menghubungkan aplikasi sistem kendali dapat dilihat pada **Kode Program 5.3** berikut:

```
1   MQTTc = MQTT.Client("serverclient",clean_session=False)
2   MQTTc.connect("localhost",1883)
```

Kode Program 5.3 Implementasi Koneksi Aplikasi dengan Broker

Aplikasi yang dibuat adalah menggunakan bahasa pemrograman python penjelasan dari **Kode Program 5.3** adalah sebagai berikut:

1. Baris 1 adalah inisialisasi aplikasi sebagai *MQTT* client.
2. Baris ke 2 adalah perintah koneksi *MQTT* client ke broker, dikarenakan aplikasi dengan broker berada pada komputer yang sama maka untuk alamat dari broker ditulis dengan localhost.

5.2.3 Implementasi *Publish*/*Subscribe* pada Sistem

Setelah semua perangkat terhubung dengan broker maka perlu ditentukan peranan dari masing – masing perangkat dalam hal pengiriman pesan, karena protokol yang digunakan dalam pertukaran pesan adalah protokol *MQTT* yang berbasis *publish*/*subscribe*, maka peran yang dilakukan oleh perangkat adalah sebagai *publisher*, *subscriber* atau keduanya, penentuan peran *publish*/*subscribe* disesuaikan dengan perancangan sebelumnya yang dapat dilihat pada **Tabel 4.1**, untuk implementasi mikrokontroler yang terintegerasi dengan lampu LED *subscriber* dapat dilihat pada **Kode Program 5.4** berikut:

```
1  client.setCallback(callback);
2  client.subscribe("1");
3  client.subscribe("2");
4  void callback(char* topic, byte* payload, unsigned int length) {
5    Serial.print("Message arrived [");
6    Serial.print(topic);
7    Serial.print("] ");
8    for (int i = 0; i < length; i++) {
9      Serial.print((char)payload[i]);
10   }
11   Serial.println();
12
13   if (strcmp(topic,"1")==0) {
14     if((char)payload[0] == '1'){
15     digitalWrite(D5, LOW);
16     Serial.println("nyala LED 1");
17     }
18     else{
19       digitalWrite(D5, HIGH);
20      Serial.println("mati LED 1");
21      }
22 }
23   if(strcmp(topic,"2")==0) {
24       if((char)payload[0] == '1')
25       {
26     digitalWrite(D7, LOW);
27     Serial.println("nyala LED 2");}
28       else{
29      digitalWrite(D7, HIGH);
30      Serial.println("mati LED 2");
31         }
32   }
33   }
```

Kode Program 5.4 Implemtasi Peran *Subscribe* pada Mikrokontroler yang Terintegrasi dengan Lampu LED

Mikrokontroler yang terintegerasi dengan lampu LED hanya akan menjalankan peran sebagai *subscriber* yaitu hanya menerima pesan dari broker tanpa melakukan pengiriman pesan ke broker, penjelasan dari **Kode Program 5.4** adalah sebagai berikut:

1. Baris ke 1 menginisialisasi method callback, method callback adalah method yang akan dijalakan ketika *subscriber* menerima pesan dari broker.
2. Baris 2-3 untuk melakukan *subscriber* pada topik 1 dan topik 2.
3. Baris 4-33 adalah implementasi method callback yang berisi seleksi perintah untuk menyalakan dan mematikan lampu LED yang terintegerasi dengan mikrokontroler.

Untuk Implementasi peran dari mikrokontroler yang terintegerasi dengan sensor DHT 11 dapat dilihat pada **Kode Program 5.5** berikut:

```
1  client.publish("suhu",temp);
```

Kode Program 5.5 Implemtasi Peran *Publish* pada Mikrokontroler yang Terintegrasi dengan DHT11

Mikrokontroler yang terintegerasi dengan DHT 11 hanya berperan sebagai *publisher* yaitu mengirimkan data yang telah didapat dari sensor DHT 11 kepada broker, penjelasan dari **Kode Program 5.5** adalah sebagai berikut:

1. pada baris 1 adalah perintah untuk mem*publish* pesan pada broker dengan parameter “suhu ” sebagai topik dari pesan yang di *publish* dan temp sebagai data sensor yang dikirimkan pada broker.

Untuk Implementasi peran dari mikrokontroler yang terintegerasi dengan sensor LDR dapat dilihat pada **Kode Program 5.6** berikut:

```
1  client.publish("cahaya",light);
```

Kode Program 5.6 Implemtasi Peran *Publish* pada Mikrokontroler yang Terintegrasi dengan Sensor LDR

Sama seperti mikrokontroler yang terintegerasi dengan DHT 11, mikrokontroler yang terintegerasi dengan sensor LDR hanya berperan sebagai *publisher*, penjelasan dari **Kode Program 5.6** adalah sebagai berikut:

1. Baris 1 adalah perintah untuk mem*publish* pesan dengan “cahaya” sebagai parameter topik, dan light sebagai hasil data sensor yang telah didapat.

Untuk Implementasi peran dari aplikasi sistem kendali dapat dilihat pada **Kode Program 5.7** berikut:

```
1  MQTTc.on_message =  on_message
2  MQTTc.subscribe("suhu")
3  MQTTc.subscribe("cahaya")
4  def on_message(MQTTc,obj,msg):
5      global LED1
6      global LED2
7      global param
8      datasuhu = r.lrange("suhu",-1,-1)
9      datacahaya = r.lrange("cahaya",-1,-1)
10     print "Telah Diterima message : "+msg.payload+" topik
11 "+msg.topic
12     r.rpush(msg.topic,msg.payload)
13     if param=="default":
14         if msg.topic=="suhu":
15             if float(msg.payload)<=paramsuhu :
16                 MQTTc.publish("1","1",qos=0,retain=False)
17                 LED1="ON"
18             elif float(msg.payload)>paramsuhu:
19                 MQTTc.publish("1","0",qos=0,retain=False)
20                 LED1="OFF"
21         elif msg.topic=="cahaya" :
22             if float(msg.payload) <=paramcahaya:
23                 MQTTc.publish("2","1",qos=0,retain=False)
24                 LED2="ON"
25             elif float(msg.payload)>paramcahaya:
26                 MQTTc.publish("2","0",qos=0,retain=False)
27                 LED2="OFF"
28
29     elif param =="auto1":
30         if msg.topic=="suhu":
31             if float(msg.payload)<=paramsuhu :
32                 MQTTc.publish("1","1",qos=0,retain=False)
33                 LED1="ON"
34             elif float(msg.payload)>paramsuhu:
35                 MQTTc.publish("1","0",qos=0,retain=False)
36                 LED1="OFF"
37
38     elif param=="auto2":
39         if msg.topic=="cahaya" :
40             if float(msg.payload) <=paramcahaya:
41                 MQTTc.publish("2","1",qos=0,retain=False)
42                 LED2="ON"
43             elif float(msg.payload)>paramcahaya:
44                 MQTTc.publish("2","0",qos=0,retain=False)
```

```
45                      LED2="OFF"
46      else :
47          print
```

Kode Program 5.7 Implemtasi Peran *Publish/Subscribe* pada Aplikasi Sistem Kendali

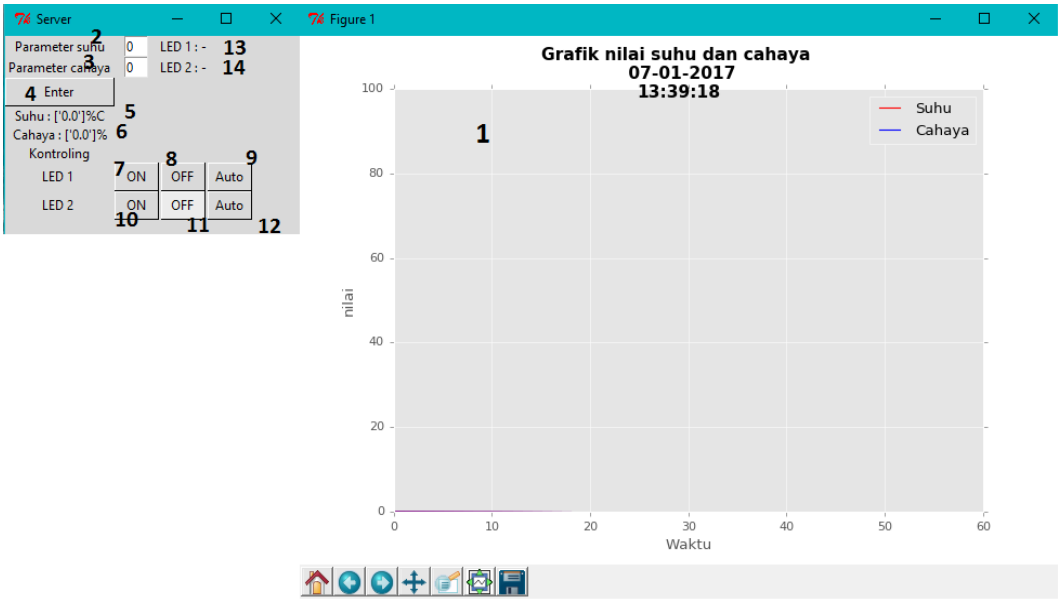
Aplikasi sistem kendali menjalakan 2 peran yaitu sebagai *publisher* dan *subscriber*, penjelasan **Kode Program 5.7** adalah sebagai berikut:

1. Baris 1 menginisialisasi method on_message, method on_message sama seperti method callback pada mikrokontroler inisialisasi ini dilakukan untuk menetukan method mana yang akan dijalankan ketika aplikasi menerima data dari broker.
2. Baris ke 2-3 untuk melakukan *subscribe* pada topik suhu dan topik cahaya, topi disesuaikan dengan pesan yang ingin diterima.
3. Baris 4-47 adalah implementasi dari method on_message berisi seleksi untuk pesan yang akan di*publish* untuk menyalakan dan mematikan lampu LED. Sedangkan untuk melakukan perannya sebagai *publisher* aplikasi sistem kendali menggunakan perintah *publish* salah satunya seperti seperti pada baris 16.

5.3 Implementasi Perangkat Lunak

5.3.1 Implementasi Aplikasi Sistem Kendali

Pada implementasi aplikasi sistem kendali dibuat aplikasi sebagai sarana untuk pengguna mengendalikan sistem, aplikasi sistem kendali yang dibuat menggunakan bahasa pemrograman python, hasil dari implementasi aplikasi sistem kendali dapat dilihat pada **Gambar 5.4** berikut:



Gambar 5.4 Implementasi Aplikasi Sistem Kendali

Hasil dari implementasi aplikasi sistem kendali dijelaskan sebagai berikut:

1. Nomor 1 menunjukan grafik yang akan menampilkan nilai dari data sensor yaitu data cahaya dan data suhu.
2. Nomor 2 adalah label dan box input parameter suhu untuk pengguna mengatur parameter nilai suhu.
3. Nomor 3 adalah label dan box input parameter cahaya untuk pengguna mengatur parameter nilai cahaya.
4. Nomor 4 adalah tombol Enter untuk mengeksekusi perintah mengganti nilai parameter suhu dan parameter cahaya berdasarkan dengan nilai yang berada diinput box parameter suhu dan parameter cahaya.
5. Nomor 5 adalah label yang digunakan untuk menampilkan nilai dari data suhu yang diterima saat itu.
6. Nomor 6 adalah label yang digunakan untuk menampilkan nilai dari data cahaya yang diterima saat itu.
7. Nomor 7 adalah tombol ON yang digunakan untuk menyalakan lampu LED 1.
8. Nomor 8 adalah tombol OFF yang digunakan untuk mematikan lampu LED 1.
9. Nomor 9 adalah tombol auto yang digunakan untuk mematikan dan menyalakan lampu LED 1 secara otomatis sesuai dengan parameter suhu dan cahaya yang telah ditentukan.
10. Nomor 10 adalah tombol ON yang digunakan untuk menyalakan lampu LED 2.
11. Nomor 11 adalah tombol OFF yang digunakan untuk mematikan lampu LED 1.
12. Nomor 12 adalah tombol auto yang digunakan untuk mematikan dan menyalakan lampu LED 2 secara otomatis sesuai dengan parameter suhu dan cahaya yang telah ditentukan.
13. Nomor 13 adalah label yang digunakan untuk menampilkan state/kondisi lampu LED 1.
14. Nomor 14 adalah label yang digunakan untuk menampilkan state/kondisi lampu LED 2.

5.3.2 Implementasi Database

Implementasi database dilakukan untuk membuat 2 buah database untuk menyimpan data sensor yang diterima, dalam pembuatan sistem ini yang digunakan adalah database redis yang menggunakan tipe data *list* yang mendukung data duplikat tipe data list ini memiliki sistem key – value. Pembuatan database redis ditulis dengan bahasa pemrograman python yang dilakukan oleh

aplikasi sistem kendali, untuk implementasi database yang digunakan dapat dilihat pada **Kode Program 5.8** berikut:

```
1  r = redis.Redis("localhost",6379)
2  r.rpush(msg.topic,msg.payload)
```

Kode Program 5.8 Implementasi Database

Untuk membuat database pada redis diperlukan untuk terkoneksi terlebih dahulu pada database redis, penjelasan dari **Kode Program 5.8** adalah sebagai berikut:

1. Baris 1 pada progam adalah perintah untuk membuat konkesi dengan database redis yang berada di localhost.
2. Baris 2 adalah perintah untuk membuat database redis bertipe list dengan menggunakan parameter topik dari pesan yang diterima sebagai key yang berarti akan membuat database dengan key “suhu” dan “cahaya” sesuai dengan topik yang di*subscribe* oleh aplikasi, sementara isi dari value yang digunakan adalah payload atau isi pesan yang diterima oleh aplikasi yaitu isi data sensor DHT11 atau sensor LDR.

5.3.3 Implementasi Monitoring Suhu dan Intensitas Cahaya dari Sensor

Implementasi monitroing suhu dan intensitas cahaya dibuat agar pengguna dapat melakukan monitoring terhadap lingkungan sekitar dilihat dari nilai sensor suhu dan sensor intensitas cahaya, untuk implementasi monitoring ini yang pertama dilakukan adalah mengimplementasikan pengiriman data dari sensor pada broker, untuk implementasi pengiriman data sensor pada broker dapat dilihat pada **Kode Program 5.9** dan **Kode Program 5.10** berikut:

```
1  float t = dht.readTemperature();
2   if (isnan(t) ) {
3      Serial.println("Failed to read from DHT sensor!");
4      return;
5    }
6    temp_str = String(t);
7    temp_str.toCharArray(temp, temp_str.length() + 1);
8    Serial.print("Temperature: ");
9    Serial.print(t);
10   Serial.print(" *C ");
11
12   Serial.print(" %\t");
13   Serial.println();
14   Serial.print("Publish message: ");
15   Serial.println(t);
16   client.publish("suhu",temp);
17
18  delay(1000);
```

Kode Program 5.9 Implementasi Pengiriman Data Suhu Ke Broker

Pada **Kode Program 5.9** menunjukan *script* yang ada pada mikrokontroler untuk mengambil data sensor DHT 11 dan mengirimkannya pada broker, penjelasan dari **Kode Program 5.9** adalah sebagai berikut:

1. Baris 1 adalah untuk mengambil data temprature pada sensor DHT11.
2. Baris 2-5 untuk melakukan pengecekan nilai sensor apakah nilai didapatkan atau tidak.
3. Baris 6-7 digunakan untuk melakukan *pharsing* tipe data dari data sensor menjadi chararray sehingga dapat dikirimkan menggunakan protokol *MQTT*.
4. Baris 8-15 untuk melakukan print pengecekan data pada serial mikrokontroler.
5. Baris 16 untuk mem*publish*/mengirimkan pesan pada broker dengan topik "suhu" dan data sensor yang disimpan pada variabel temp.
6. Baris 18 untuk mengatur jeda pengambilan dan pengiriman data sensor suhu selama 1000ms.

```
1    Serial.println(light_sensor.getPercentValue());
2    l=light_sensor.getPercentValue();
3    light_str = String(l);
4    light_str.toCharArray(light, light_str.length() + 1);
5    client.publish("cahaya",light);
6   delay(1000);
```

Kode Program 5.10 Implementasi Pengiriman Data Intensitas Cahaya Ke Broker

Pada **Kode Program 5.10** adalah script pada mikrokontroler yang terintegrasi dengan sensor LDR yang digunakan untuk mengambil data dari sensor LDR kemudian mengirimkannya pada broker, penjelasan dari **Kode Program 5.10** adalah sebagai berikut:

1. Baris 1 melakukan print data hasil tangkapan sensor LDR pada serial mikrokontroler.
2. Baris 2 menyimpan data hasil tangkapan sensor LDR pada variabel l.
3. Baris 3-4 untuk melakukan *pharsing* tipe data menjadi chararray agar dapat dikirimkan menggunakan protokol *MQTT*.
4. Baris ke 5 untuk mem*publish*/mengirimkan pesan pada broker dengan topik "cahaya" dan data sensor yang disimpan pada variabel light,
5. baris ke 6 untuk mengatur jeda pengambilan dan pengiriman data sensor suhu selama 1000ms.

Selanjutnya setelah berhasil mengirimkan data pada broker, broker selanjutnya akan secara otomatis mengirimkan pesan kepada aplikasi yang melakukan *subscribe* topik suhu dan topik cahaya, untuk membuat sistem monitoring diperlukan untuk menyimpan data sensor pada database redis, implementasi penyimpanan data pada databse redis dapat dilihat pada **Kode Program 5.8**, karena syntak yang digunakan untuk membuat database redis dan menyimpan database pada redis sama dengan menggunakan sistem key-value seperti yang telah dijelaskan sebelumnya. Setelah menyimpan data pada databse yang selanjutnya yang dilakukan adalah implementasi untuk menampilkan data pada grafik, untuk implementasi menampilkan data pada grafik dapat dilihat pada **Kode Program 5.11** berikut:

```
1   def graph(self):
2           suhu1 = r.lrange("suhu",-1,-1)
3           cahaya1 = r.lrange("cahaya",-1,-1)
4           date = datetime.now().second
5           date1 = datetime.now().strftime("%d-%m-%Y")
6           date2 = datetime.now().strftime("%H:%M:%S")
7           join1="".join(suhu1)
8           suhugraph=float(join1)
9           join2="".join(cahaya1)
10          cahayagraph=float(join2)
11          plt.ion()
12          plt.xlabel("Waktu")
13          plt.ylabel("nilai")
14          plt.axis([0, 60, 0, 100])
15          if date==00 :
16              plt.cla()
17              plt.axis((0, 60, 0, 100))
18              self.x_list_1 = []
19              self.y_list_1 = []
20              self.x_list_2 = []
21              self.y_list_2 = []
22          plt.suptitle("Grafik nilai suhu dan cahaya \n
23  "+date1+"\n"+date2, fontsize=14, fontweight='bold')
24          self.x_list_1.append(date)
25          self.y_list_1.append(suhugraph)
26          self.x_list_2.append(date)
27          self.y_list_2.append(cahayagraph)
28          plt.legend(['Suhu','Cahaya'])
29          plt.plot(self.x_list_1, self.y_list_1, "r-")
30          plt.plot(self.x_list_2, self.y_list_2, "b-")
31          self.after(1000, self.graph)
```

Kode Program 5.11 Implementasi Menampilkan Data pada Grafik

Untuk script menampilkan data pada grafik dijelaskan sebagai berikut:

1. Baris 1 adalah inisilisasi method graf yang akan digunakan untuk menampilkan grafik yang berisi data sensor.
2. Baris 2-3 untuk mengambil data dari database redis, baris ke 2 untuk mengambil nilai terakhir pada database redis yang memiliki key “suhu” dan baris ke 3 untuk mengambil nilai terakhir pada database redis yang memiliki key “cahaya”.
3. Baris 4-6 digunakan untuk mengambil nilai waktu saat ini, baris 4 mengambil nilai detik, baris 5 mengambil nilai hari,bulan, dan tahun saat ini, baris 5, baris 6 untuk mengambil nilai jam,menit dan detik.
4. Baris 7-10 digunakan untuk melakukan pharsing tipe data untuk nilai data suhu agar dapat ditampilkan pada grafik.
5. Baris 11 digunakan untuk menampilkan grafik interkatif.
6. Baris 12-13 digunakan untuk menampilkan label pada axis X dan Y pada grafik.
7. Baris 14 digunakan digunakan untuk mengatur skala dari axis X dan Y.
8. Baris 15-21 digunakan sebagai perintah seleksi ketika detik waktu menunjukan 00 maka grafik akan dibersihkan.
9. Baris 22 untuk menambahkan header pada grafik.

10. Baris 24-27 memasukan nilai data sensor dan waktu pada masing – masing array.
11. Baris 28 untuk menambhakan legenda pada grafik.
12. Baris 29-30 untuk menggambar nilai dari data sensor pada grafik.
13. Baris 31 digunakan untuk menjalakan method graph setiap satu detik yang berfungsi untuk mengupdate nilai dari grafik yang ada.

Setelah menampilkan data pada grafik sesuai dengan perancangan yang dibuat, selanjut dilakukan implmentasi menampilkan data suhu dan cahaya pada label di dalam aplikasi, implementasi untuk menampilkan data pada label aplikasi dapat dilihat pada **Kode Program 5.12** berikut:

```
1  self.L5['text'] = "Suhu : " +  str(r.lrange("suhu",-1,-1))+"%C"
2  self.L6['text'] = "Cahaya : " +  str(r.lrange("cahaya",-1,-1))+"%"
3  self.after(1000, self.update)
```

Kode Program 5.12 Implementasi Menampilkan Data pada Aplikasi

Penjelasan dari **Kode Program 5.12** adalah sebagai berikut:

1. Baris 1-2 digunakan untuk merubah text pada label L5 dan L6 dengan nilai data sensor yang di ambil dari database, script ini terdapat pada method update.
2. Baris 3 digunakan perintah untuk menjalakan method update setiap 1000ms agar data yang ditampilkan realtime.

5.3.4 Implementasi Kontroling Lampu LED

Implementasi kontroling lampu LED dilakukan untuk pengguna agar dapat mengatur lampu LED menyala atau mati secara manual dengan menekan tombol ON/OFF pada aplikasi atau mengaturnya menyala atau mati secara otomatis sesuai dengan parameter yang ditentukan, seperti yang telah dijelaskan sebelumnya terdapat 2 buah lampu LED masing – masing dari lampu LED memiliki tombol ON,OFF dan auto sendiri seperti yang ditunjukan sebelumnya pada **Gambar 5.4**, untuk implementasi menyalakan lampu LED 1 dapat dilihat pada **Kode Program 5.13** berikut:

```
1  MQTTc.publish("1","1",qos=0,retain=False)
2  if param=="auto2" or param=="default":
3      param="auto2"
4  else :
5      param="on1"
6  LED1="ON"
```

Kode Program 5.13 Implementasi Menyalakan Lampu LED 1

Untuk menyalakan lampu LED 1 pengguna perlu menekan tombol ON yang sejajar pada label LED 1, penjelasan dari **Kode Program 5.13** adalah sebagai berikut:

1. Baris 1 script aplikasi akan melakukan *publish* pesan dengan topik 1, dan isi pesan 1 pada broker.

2. Baris 2-5 aplikasi akan mengantur nilai parameter param yang digunakan sebagai parameter seleksi pada method on_message sehingga lampu LED1 dapat tetap menyala, dan aplikasi tetap menerima data sensor.
3. Baris ke 6 mengatur nilai LED1 dengan ON.

Proses yang terjadi untuk menyalakan LED 2 juga sama seperti proses untuk menyalakan LED 1, untuk implementasi menyalakan lampu LED 2 dapat dilihat pada **Kode Program 5.14** berikut:

```
1  MQTTc.publish("2","1",qos=0,retain=False)
2  if param=="auto1" or param=="default":
3      param="auto1"
4  else :
5      param="on2"
6  LED2="ON"
```

Kode Program 5.14 Implementasi Menyalakan Lampu LED 2

Untuk menyalakan LED 2 proses yang terjadi sama seperti menyalakan LED 1 yaitu pengguna perlu menekan tombol ON yang sejajar dengan label LED 2, penjelasan dari **Kode Program 5.14** adalah sebagai berikut:

1. Baris 1 akan mem*publish* pesan untuk menyalakan lampu LED 2.
2. Baris 2-5 mengatur parameter param.
3. Baris 6 untuk mengatur nilai LED2 menjadi ON.

Setelah dapat menyalakan lampu LED diperlukan juga untuk sistem dapat mematikan lampu LED 1 dan lampu LED 2, untuk implementasi mematikan lampu LED 1 dapat dilihat pada **Kode Program 5.15** berikut:

```
1  MQTTc.publish("1","0",qos=0,retain=False)
2  if param=="auto2" or param=="default":
3      param="auto2"
4  else :
5      param="off1"
6  LED1="OFF"
```

Kode Program 5.15 Implementasi Mematikan Lampu LED 1

Sama seperti halnya menyalakan lampu LED, proses mematikan dimulai saat pengguna menekan tombol OFF yang berada sejajar pada label LEDnya, untuk mematikan lampu LED 1 diperlukan untuk menekan tombol OFF yang sejajar pada label LED 1 kemudian akan menjalakan script diatas penjelasan dari **Kode Program 5.15** adalah sebagai berikut:

1. Baris 1 akan mem*publish* pesan yang berisi perintah untuk mematikan lampu LED 1.
2. Baris 2-5 untuk mengatur nilai dari parameter param.
3. Baris 6 untuk memberi niliai OFF pada variabel LED1.

Untuk mematikan lampu LED 2 proses yang terjadi hampir sama, untuk implementasi mematikan lampu LED 2 dapat dilihat pada **Kode Program 5.16** berikut:

```
1  MQTTc.publish("2","0",qos=0,retain=False)
2  if param=="auto1" or param=="default":
```

```
3       param="auto1"
4   else :
5       param="off2"
6   LED2="OFF"
```

Kode Program 5.16 Implementasi Mematikan Lampu LED 2

Untuk proses mematikan lampu LED 2, proses yang terjadi hampir sama dengan proses menyalakan ataupun mematikan lampu LED yang sebelumnya telah dijelaskan, pengguna harus menakan tombol OFF yang sejajar dengan label LED 2 penjelasan dari **Kode Program 5.16** adalah sebagai berikut:

1. baris ke 1 adalah perintah untuk aplikasi mem*publish* pesan untuk mematikan lampu LED 2 seperti pada.
2. baris 2-5 untuk mengatur nilai parameter param.
3. baris ke 6 untuk memberi nilai OFF pada variabel LED2.

Setelah dapat menyalakan dan mematikan lampu LED dengan menekan tombol yang ada pada aplikasi sistem kendali, aplikasi yang dibuat juga harus dapat mematikan dan menyalakan lampu LED secara otomatis berdasarkan nilai dari data sensor, ada beberapa tipe untuk mengatur menyalakan dan mematikan lampu LED secara otomatis, yaitu secara default ketika aplikasi dinyalakan aplikasi akan dengan otomatis menyalakan dan mematikan lampu LED sesuai dengan nilai data sensor yang dibandingkan dengan nilai parameternya, kemudian jika pengguna hanya mengingkan salah satu lampu LED saja yang menyala atau mati berdasarkan nilai sensor, ada 2 buah tombol auto, masing – masing untuk LED1 dan LED2 tombol auto ini berfungsi untuk mengatur agar lampu LED yang dimaksud menyala dan mati secara otomatis. Untuk implementasi default menyalakan dan mematikan lampu LED secara otomatis dapat dilihat pada **Kode Program 5.17** berikut:

```
1   param="default"
    if param=="default":
2       if msg.topic=="suhu":
3           if float(msg.payload)<=paramsuhu :
4               MQTTc.publish("1","1",qos=0,retain=False)
5               LED1="ON"
6           elif float(msg.payload)>paramsuhu:
7               MQTTc.publish("1","0",qos=0,retain=False)
8               LED1="OFF"
9       elif msg.topic=="cahaya" :
10          if float(msg.payload) <=paramcahaya:
11              MQTTc.publish("2","1",qos=0,retain=False)
12              LED2="ON"
13          elif float(msg.payload)>paramcahaya:
14              MQTTc.publish("2","0",qos=0,retain=False)
15              LED2="OFF"
16
```

Kode Program 5.17 Implementasi Default Menyalakan dan Mematikan Lampu LED Secara Otomatis

Penjelasan dari **Kode Program 5.17** adalah sebagai berikut:

1. Baris ke 1 adalah perintah memeberi nilai “default” untuk parameter param seperti yang terlihat ketika aplikasi dijalankan aplikasi.

2. Baris 2-16 terletak pada method on_message pada aplikasi yaitu method yang akan dijalankan ketika aplikasi menerima pesan dari broker, saat menerima pesan dengan topik “suhu” pada baris ke 5, maka aplikasi akan membandingkan nilai payload dalam hal ini nilai payload adalah nilai dari sensor suhu akan dibandingkan dengan nilai paramsuhu yaitu nilai parameter suhu, ketika nilai data sensor suhu <= nilai parameter suhu maka aplikasi akan mem*publish* pesan untuk menyalakan lampu LED 1 seperti yang terlihat pada baris 4-6, dan ketika nilai data sensor suhu > dari nilai parameter suhu maka aplikasi akan mem*publish* pesan untuk mematikan lampu LED 1. Dan ketika aplikasi menerima pesan dengan topik “cahaya” seperti pada baris 10, aplikasi juga akan melakukan hal yang sama yaitu akan membandingan nilai dari data sensor cahaya dengan nilai parameter suhu ketika nilai data sensor cahaya <= nilai parameter cahaya maka aplikasi akan mem*publish* pesan untuk menyalakan lampu LED 2 dengan perintah pada baris 11-13, dan akan mem*publish* pesan untuk mematikan lampu LED 2 ketika nilai data “cahaya” > dari nilai parameter cahaya, dengan perintah seperti pada baris 14-16.

Menyalakan dan mematikan lampu LED juga dapat dilakukan dengan menekan tombol auto, untuk implementasi menyalakan dan mematikan lampu LED dengan menekan tombol auto dapat dilihat pada **Kode Program 5.18** dan **Kode Program 5.19** berikut:

```
1   if param=="auto2" or param == "default":
2       param="default"
3   else:
4       param="auto1"
5
6   elif param =="auto1":
7       if msg.topic=="suhu":
8           if float(msg.payload)<=paramsuhu :
9               MQTTc.publish("1","1",qos=0,retain=False)
10              LED1="ON"
11          elif float(msg.payload)>paramsuhu:
12              MQTTc.publish("1","0",qos=0,retain=False)
13              LED1="OFF"
```

Kode Program 5.18 Implementasi Menyalakan dan Mematikan Lampu LED Secara Otomatis dengan Menekan Tombol Auto yang Sejajar dengan Label LED 1

Untuk mengatur lampu LED menyala atau mati sesuai dengan nilai sensor data suhu yang diterima oleh aplikasi pengguna perlu menekan tombol auto yang berada sejajar pada label LED 1, ketika tombol ditekan script pada **Kode Program 5.18** akan dijalankan, penjelasan dari **Kode Program 5.18** adalah sebagai berikut:

1. Baris 1-4 ketika tombol auto ditekan maka aplikasi akan mengatur nilai dari parameter param sesui degan kondisi nilai paramter param saat itu, ketika nilai paramater param menjadi “default” maka script pada **Kode Program 5.17** akan dijalankan.
2. Baris 6-13 ketika parameter param bernilai auto 1 maka kemudian script pada baris 6-13 akan dijalankan. Baris 6-13 adalah script yang berada

dimethod on_message pada aplikasi yang akan mengirimkan perintah menyalakan atau mematikan lampu LED 1 sesuai dengan nilai data suhu yang diterima oleh aplikasi.

```
1   if param=="auto1" or param=="default":
2       param="default"
3   else:
4       param="auto2"
5
6   elif param=="auto2":
7       if msg.topic=="cahaya" :
8           if float(msg.payload) <=paramcahaya:
9               MQTTc.publish("2","1",qos=0,retain=False)
10              LED2="ON"
11          elif float(msg.payload)>paramcahaya:
12              MQTTc.publish("2","0",qos=0,retain=False)
13              LED2="OFF"
```

Kode Program 5.19 Implementasi Menyalakan dan Mematikan Lampu LED Secara Otomatis dengan Menekan Tombol Auto yang Sejajar dengan Label LED 2

Kode Program 5.19 adalah script yang digunakan untuk menyalakan dan mematikan lampu LED 2 secara otomatis dengan menkan tombol auto yang berada sejajar dengan label LED 2 pada aplikasi, penjelasan dari **Kode Program 5.19** adalah sebagai berikut:

1. Baris 1-4 akan mengatur nilai parameter param sesuai dengan kondisi saat tombol ditekan, kemudian nilai parameter param akan menjadi paramater untuk seleksi pada method on_message.
2. Baris 6-13 ketika parameter param bernilai auto 1 maka kemudian script pada baris 6-13 akan dijalankan. Baris 6-13 adalah script yang berada dimethod on_message pada aplikasi yang akan mengirimkan perintah menyalakan atau mematikan lampu LED 1 sesuai dengan nilai data suhu yang diterima oleh aplikasi.

Setelah melakukan implmentasi pada aplikasi sistem kendali untuk melakukan kontroling lampu LED diperlukan juga implementasi kode program pada mikrokontroler yang terintegrasi dengan lampu LED, untuk implementasi pada mikrokontroler yang terintegerasi dengan lampu LED dapat dilihat pada **Kode Program 5.20** berikut:

```
1    if (strcmp(topic,"1")==0) {
2      if((char)payload[0] == '1'){
3      digitalWrite(D5, LOW);
4      Serial.println("nyala LED 1");
5      }
6      else{
7        digitalWrite(D5, HIGH);
8       Serial.println("mati LED 1");
9       }
10   }
11   if(strcmp(topic,"2")==0) {
12       if((char)payload[0] == '1')
13       {
14     digitalWrite(D7, LOW);
15     Serial.println("nyala LED 2");}
16       else{
17      digitalWrite(D7, HIGH);
18      Serial.println("mati LED 2");
19         }
20   }
```

Kode Program 5.20 Implementasi Menyalakan dan Mematikan Lampu LED pada Mikrokontroler

Script **Kode Program 5.20** adalah script pada mikrokontroler yang terintegerasi dengan lampu LED, script diatas bertujuan untuk melakukan seleksi pesan dari broker, pesan nantinya akan diseleksi untuk menentukan menyala atau matinya lampu LED, penjelasan dari **Kode Program 5.20** adalah seagai berikut:

1. Baris 1-10 adalah seleksi untuk lampu LED 1 ketika topik yang diterima adalah 1 dan nilai pesan yang diterima juga 1 maka mikrokontroler akan menyalakan lampu LED 1, ketika isi pesan bernilai 0 maka mikrokontroler akan mematikan lampu LED.
2. Baris 11-12 digunakan untuk seleksi lampu LED 2. Implementasi yang terakhir untuk kontroling LED adalah mengatur nilai parameter suhu dan nilai paramater cahaya dengan input dari user.

Untuk implementasi mengatur nilai parameter suhu dan nilai parameter cahaya dapat dilihat pada **Kode Program 5.21** berikut:

```
1  global paramsuhu
2  global paramcahaya
3  self.CheckVar1 = self.E1.get()
4  self.CheckVar2 = self.E2.get()
5  paramsuhu=float(self.CheckVar1)
6  paramcahaya=float(self.CheckVar2)
```

Kode Program 5.21 Implementasi Input Nilai Parameter Suhu dan Paramater Cahaya

Untuk mengatur nilai parameter suhu dan nilai paramaeter cahaya pengguna perlu untuk menginpukan nilai pada input box yang pada aplikasi seperti yang terlihat pada **Gambar 5.4** dimana ada 2 input box yaitu input box untuk nilai parameter suhu dana input box untuk nilai parameter cahaya, setelah menginputkan nilai pada input pengguna harus menekan tombol Enter yang berada di bawah input box, setelah tombol Enter di tekan maka script pada **Kode Program 5.21** akan dijalankan, penjelasan dari **Kode Program 5.21**

1. Baris 3-4 akan mengambil nilai dari input box.
2. Baris 5-6 memberikan nilai dari inputbox ke variabel paramsuhu dan variabel paramcahaya.
3. baris 1-2 dibutuhkan deklrasi variabel global karena akan merubah nilai variabel global.

5.3.5 Implementasi Monitoring Lampu LED

Setelah melakukan implementasi kontroling lampu LED selanjut dilakukan implementasi monitoring lampu LED, yang dimaksud dengan monitoring lampu LED adalah menampilakan kondisi atau state LED 1 dan LED 2 pada aplikasi, pengimplementasian monitoring lampu LED menggunakan label yang ditampilkan pada user interface dari aplikasi sistem kendali, pengimplementasian monitoring lampu LED dapat dilahat pada **Kode Program 5.22** berikut:

```
1  LED1 ='-'
2  LED2='-'
3  self.L3 =Label(self,text="LED 1 : "+LED1)
4  self.L4 =Label(self,text="LED 2 : "+LED2)
5  self.L3['text'] = "LED 1 : " + LED1
6  self.L4['text'] = "LED 2 : " + LED2
```

Kode Program 5.22 Implementasi Monitoring Lampu LED

Pada **Kode Program 5.22** hanya memperlihatkan bagaimana untuk memasukan data variabel LED1 dan LED2 pada label yang selanjutnya label akan di tampilkan pada user interfacae dari aplikasi sistem kendali, penjelasan dari **Kode Program 5.22** adalah sebagai berikut:

1. Baris 1-2 adalah inisialisai variabel LED1 dan LED2.
2. Baris 3-4 inisialisasi label L3 dan L4, baris ke 5-6 adalah inisialisasi untuk merubah nilai text dari L3 dan L4.

Sementara proses perubahan nilai variabel LED1 dan LED2 dapat dilihat pada **Kode Program 5.23** berikut:

```
1  if msg.topic=="suhu":
2            if float(msg.payload)<=paramsuhu :
3                if LED1 =="ON"or LED1=="-":
4                  LED1="ON"
5                elif LED1=="OFF":
6                  MQTTc.publish("1","1",qos=0,retain=False)
7                  LED1="ON"
8            elif float(msg.payload)>paramsuhu:
9                if LED1 =="OFF" or LED1=="-":
10                 LED1="OFF"
11               elif LED1=="ON":
12                 MQTTc.publish("1","0",qos=0,retain=False)
13                 LED1="OFF"
14     elif msg.topic=="cahaya" :
15         if float(msg.payload) <=paramcahaya:
16             if LED2 =="ON"or LED2=="-":
17                 LED2="ON"
18             elif LED2=="OFF":
19                 MQTTc.publish("2","1",qos=0,retain=False)
20                 LED2="ON"
21
```

```
        elif float(msg.payload)>paramcahaya:
              if LED2 =="OFF"or LED2=="-":
                 LED2="OFF"
              elif LED2=="ON":
                MQTTc.publish("2","0",qos=0,retain=False)
                LED2="OFF"

elif param =="auto1":
    if msg.topic=="suhu":
        if float(msg.payload)<=paramsuhu :
            MQTTc.publish("1","1",qos=0,retain=False)
            LED1="ON"
        elif float(msg.payload)>paramsuhu:
            MQTTc.publish("1","0",qos=0,retain=False)
            LED1="OFF"
elif param=="auto2":
    if msg.topic=="cahaya" :
        if float(msg.payload) <=paramcahaya:
            MQTTc.publish("2","1",qos=0,retain=False)
            LED2="ON"
        elif float(msg.payload)>paramcahaya:
            MQTTc.publish("2","0",qos=0,retain=False)
            LED2="OFF"
if x=="on1":
    MQTTc.publish("1","1",qos=0,retain=False)
    if param=="auto2" or param=="default":
        param="auto2"
    else :
        param="on1"
    LED1="ON"
elif x=="off1":
    MQTTc.publish("1","0",qos=0,retain=False)
    if param=="auto2" or param=="default":
        param="auto2"
    else :
        param="off1"
    LED1="OFF"
elif x=="auto1":
    if param=="auto2" or param == "default":
        param="default"
    else:
        param="auto1"
elif x=="on2":
    MQTTc.publish("2","1",qos=0,retain=False)
    if param=="auto1" or param=="default":
        param="auto1"
    else :
        param="on2"
    LED2="ON"
elif x=="off2":
    MQTTc.publish("2","0",qos=0,retain=False)
    if param=="auto1" or param=="default":
        param="auto1"
    else :
        param="off2"
    LED2="OFF"
elif x=="auto2":
    if param=="auto1" or param=="default":
        param="default"
    else:
        param="auto2"
```

Kode Program 5.23 Implementasi Merubah Nilai Variabel LED1 dan LED2

Pada **Kode Program 5.23** adalah program untuk seleksi perintah menyalakan atau mematikan lampu LED seperti yang telah dijelaskan pada sub bab sebelumya, tetapi pada proses terjebut terjadi perubahan nilai variabe LED1 dan variabel LED2 sesuai dengan kondisi dari lampu LED tersebut, nilai “ON” akan di berikan pada variabel LED1 atau variabel LED 2 ketika program melakukan *publish* perintah untuk menyalakan lampu LED 1 atau LED 2 dan nilai “OFF” akan diberikan pada variabel LED1 atau variabel LED2 ketika program melakukan *publish* perintah mematikan lampu LED 1 atau LED 2.

BAB 6 PENGUJIAN DAN ANALISA HASIL PENGUJIAN

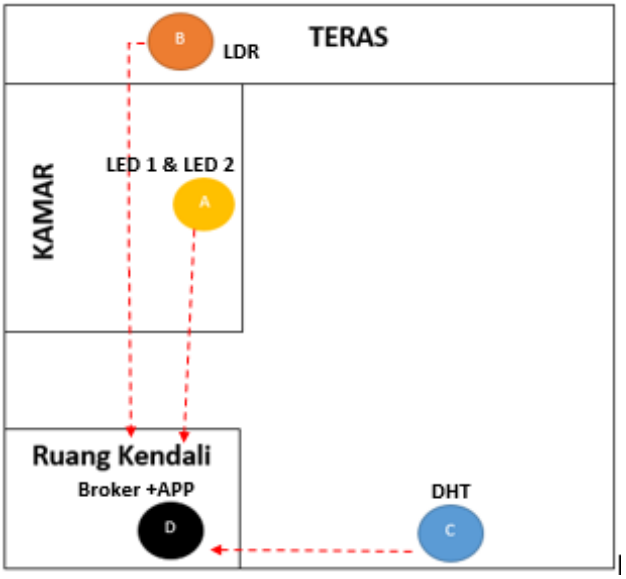
6.1 Perancangan Pengujian

Perancangan pengujian dibutuhkan untuk mengetahui bagaimana melakukan pengujian untuk selanjutnya hasil pengujian dianalisis agar dapat mengetahui kinerja dari sistem yang telah dibuat dalam perancangan pengujian ada 2 buah perancangan yang dibuat yaitu perancangan pengujian kehandalan sistem, dan pengujian fungsionalitas dari aplikasi sistem kendali yang telah dibuat. Dengan adanya pengujian diharapkan dapat menghasilkan data – data yang dapat memepresentasikan hasil dari kinerja sistem yang telah dibuat secara keseluruhan.

6.1.1 Perancangan Pengujian Kehandalan Sistem

Pengujian kehandalan sistem dibuat untuk menilai kinerja dari protokol komunikasi yang diterapkan pada sistem yaitu protokol komunikasi *MQTT*, *MQTT* yang mengatur alur dari komunikasi data diharapkan dapat membuat sistem berjalan dengan baik untuk itu dibutuhkan pengujian kehandalan sistem untuk mengetahui dan menilai hasil kinerja dari protokol komunikasi *MQTT*, pengujian kehandalan sistem dilakukan dengan cara mengamati proses pengiriman pesan dari protokol komuikasi *MQTT* dengan beberapa nilai parameter yang ditentukan.

Perancangan yang dilakukan untuk melakukan pengujian kehandalan sistem adalah pertama membuat topologi penempatan node – node yang berada di dalam sistem, topologi yang digunakan sama dengan topologi *smarthome* yang dapat dilihat pada **Gambar 4.1**¸ semua node berada ditempat terpisah diruangan yang berbeda di dalam rumah, topologi untuk perancangan dapat dilihat pada **Gambar 6.1** berikut:



Gambar 6.1 Topologi Perancangan Pengujian

Pada **Gambar 6.1** node A,B,C,D adalah node yang sama seperti yang dijelaskan pada bab perancangan, 4 node tersebut terdiri dari 3 mikrokontroler wemos dan

1 komputer server, dimana node A adalah mikrokontroler yang terintegerasi dengan lampu LED 1 dan LED 2, node B adalah mikrokontroler yang terintegerasi dengan LDR, node C merupakan mikrokontroler yang terintegerasi dengan DHT 11, kemudian node D merupakan komputer server yang terpasang dengan mosquitto broker dan aplikasi sistem kendali dan sekaligus menjadi akses point untuk jaringan yang dibentuk, garis merah putus – putus pada **Gambar 6.1** menunjukan koneksi wireless dari node A,B,D ke node C, koneksi ini adalah koneksi pembuatan jaringan diamana node D sebagai akses point akan menjembatani pembentukan jaringan, dan garis putus – putus berwarna merah juga menunjukan koneksi dari node A,B,C menuju node D, koneksi ini adalah koneksi untuk terhubung dengan mosquitto broker yang berada pada node D, setelah melakukan penempatan node sesuai dengan topologi dilakukan skenario pengaturan pengiriman pesan data sensor dari mikrontroler B menuju broker dan dari mikrokontroller C menuju broker, pada skenario pertama mikrokontroler B dan C akan mengirimkan data setiap 10 milidetik, skenario kedua mikrontroler B dan C akan mengirimkan data setiap 100 milidetik, dan skenario ke 3 mikrontroler B dan C akan mengirimkan data setiap 1000 milidetik.

Dimana kemudian setelah menjalankan skenario pengujian pada penelitian ini akan menguji integritas data dimana pada pengujian ini akan di bandingkan 50 data pertama yang diterima oleh server dan disimpan pada database redis di bandingkan oleh 50 data pertama yang di publish oleh masing – masing mikrokontroller yang terintegrasi dengan sensor. Setelah menguji integritas data pada pengujian ini pula akan di lihat nilai delta time dari masing - masing skenario pengujian. Delta time sendiri adalah lamanya jeda waktu yang di perlukan untuk penerimaan paket.

6.1.2 Perancangan Pengujian Funsionalitas Aplikasi Sistem Kendali

Agar dapat mengetahui apakah aplikasi yang telah dibuat dapat menjalakan setiap fungsinya dengan benar diperkukan sebuah pengujian untuk menguji hal tersebut. Skenario dari pengujian fungsionalitas aplikasi dijelakan pada tabel – tabel berikut:

Tabel 6.1 Skenario Pengujian Menyalakan dan Mematikan

Test Case	Pengujian Menyalakan dan mematikan lampu LED secara otomatis sesuai dengan parameter ketika aplikasi dijalankan
Prosedur	1. Pengguna Menjalakan Aplikasi ketika semua perangkat telah terhubung dengan sistem.

Keluaran yang Diharapkan	Aplikasi Menunjukan state/kondisi dari lampu LED 1 dan lampu LED 2 Kondisi lampu LED 1 dan LED 2 sesuai dengan yang tertera pada aplikasi

Tabel 6.2 Skenario Pengujian Menyalakan lampu LED 1 dengan Tombol ON

Test Case	Pengujian Menyalakan lampu LED 1 dengan menekan tombol ON yang sejajar dengan label LED 1
Prosedur	1. Pengguna menekan tombol ON yang berada sejajar dengan label LED 1
Keluaran yang Diharapkan	Aplikasi Menunjukan state/kondisi dari lampu LED 1 dengan status ON Lampu LED 1 menyala

Tabel 6.3 Skenario Pengujian Menyalakan lampu LED 2 dengan Tombol ON

Test Case	Pengujian Menyalakan lampu LED 2 dengan menekan tombol ON yang sejajar dengan label LED 2
Prosedur	1. Pengguna menekan tombol ON yang berada sejajar dengan label LED 2
Keluaran yang Diharapkan	Aplikasi Menunjukan state/kondisi dari lampu LED 2 dengan status ON Lampu LED 2 menyala

Tabel 6.4 Skenario Pengujian Mematikan lampu LED 1 dengan Tombol OFF

Test Case	Pengujian Mematikan lampu LED 1 dengan menekan tombol OFF yang sejajar dengan label LED 1
Prosedur	1. Pengguna menekan tombol OFF yang berada sejajar dengan label LED 1

Keluaran yang Diharapkan	Aplikasi Menunjukan state/kondisi dari lampu LED 1 dengan status OFF Lampu LED 1 mati

Tabel 6.5 Skenario Pengujian Mematikan lampu LED 1 dengan Tombol OFF

Test Case	Pengujian Mematikan lampu LED 2 dengan menekan tombol OFF yang sejajar dengan label LED 2
Prosedur	1. Pengguna menekan tombol OFF yang berada sejajar dengan label LED 2
Keluaran yang Diharapkan	Aplikasi Menunjukan state/kondisi dari lampu LED 2 dengan status OFF Lampu LED 2 mati

Tabel 6.6 Skenario Pengujian Menyalakan dan Mematikan lampu LED 1 Secara Otomatis dengan Tombol auto

Test Case	Pengujian menyalakan dan mematikan lampu LED 1 secara otomatis dengan menekan tombol auto yang sejajar dengan label LED 1
Prosedur	1. Pengguna menekan tombol auto yang berada sejajar dengan label LED 1
Keluaran yang Diharapkan	Aplikasi Menunjukan state/kondisi dari lampu LED 1. State atau kondisi LED 1 sesuai dengan yang terlihat pada aplikasi.

Tabel 6.7 Skenario Pengujian Menyalakan dan Mematikan lampu LED 2 Secara Otomatis dengan Tombol auto

Test Case	Pengujian menyalakan dan mematikan lampu LED 2 secara otomatis dengan menekan tombol auto yang sejajar dengan label LED 2

Prosedur	1. Pengguna menekan tombol auto yang berada sejajar dengan label LED 2
Keluaran yang Diharapkan	Aplikasi Menunjukan state/kondisi dari lampu LED 2. State atau kondisi LED 2 sesuai dengan yang terlihat pada aplikasi.

Tabel 6.8 Skenario Pengujian Menampilkan Grafik Aplikasi

Test Case	Pengujian menampilkan grafik pada aplikasi
Prosedur	1. Pengguna Menjalakan Aplikasi ketika semua perangkat telah terhubung dengan sistem.
Keluaran yang Diharapkan	Aplikasi menampilkan grafik sesuai dengan nilai pada database redis

Tabel 6.9 Skenario Pengujian Merubah Nilai Parameter Suhu dan Cahaya

Test Case	Pengujian merubah nilai parameter suhu dan cahaya
Prosedur	1. Pengguna melakukan input pada input box yang telah disediakan kemudian menekan tombol Enter
Keluaran yang Diharapkan	Aplikasi merubah nilai parameter suhu dan cahaya sesuai dengan nilai yang diinputkan pengguna.

Tabel 6.10 Skenario Pengujian Merubah Nilai Parameter suhu dan Cahaya

Test Case	Pengujian Menampilkan Nilai data sensor pada label diaplikasi
Prosedur	1. Aplikasi di jalankan
Keluaran yang Diharapkan	Aplikasi menampilkan nilai terakhir dari data sensor suhu dan data sensor cahaya yang diterima

6.2 Hasil dan Analisis Pengujian

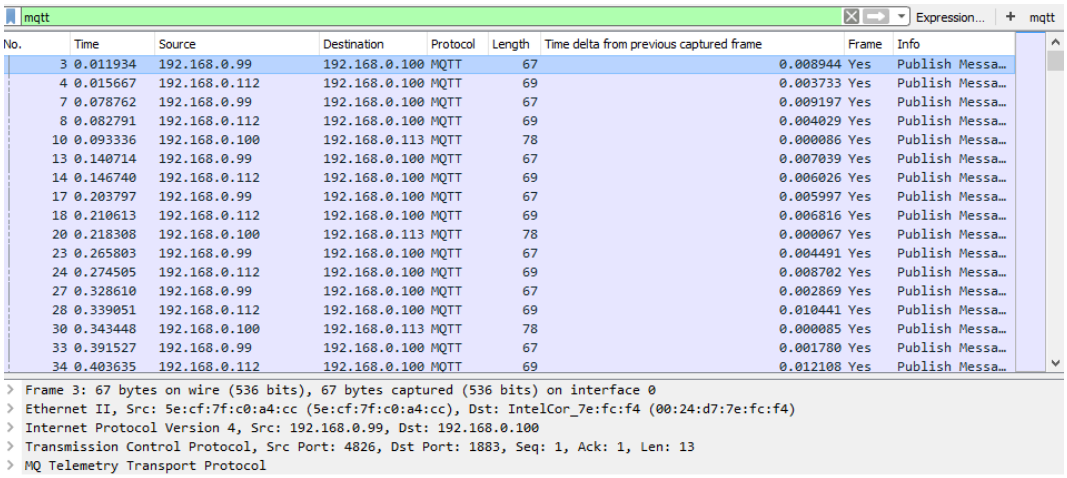
Setelah dilakukan pengujian terhadap sistem dan aplikasi maka diperlukan untuk menganalisa hasil dari pengujian tersebut, hasil analisa diperuntukan agar mengetahui kesinambungan antara hasil dari penelitian dengan tujuan penelitian yang telah ditetapkan sebelumnya, hasil dan analisis pengujian juga digunakan untuk mengetahui seberapa handal sistem yang dibuat mengacu pada parameter – parameter yang telah ditentukan yaitu integritas data dan delta time.

6.2.1 Hasil dan Analisis Pengujian Kehandalan Sistem

6.2.1.1 Pengujian Delta Time

Pengujian kehandalan sistem dilakukan sesuai dengan perancangan yang telah dibuat sebelumnya yaitu pengujian dilakukan dengan menemptakan node – node ditempat tertentu sesuai dengan topologi yang di rancang pada **Gambar 6.1** kemudian dilakukan 3 skenario berbeda yaitu jeda waktu pengiriman data dari mikrokontroler yang terintegerasi dengan sensor LDR dan DHT11 menuju broker.

Pengujian dilakukan selama kurang lebih 15 menit untuk masing – masing skenario dan ulang sebanyak 3 kali untuk masing – masing skenario, pengujian dilakukan dengan menggunakan bantuan aplikasi wireshark yang melakukan *capture* pada interface jaringan dikomputer server yang terkoneksi dengan jaringan *smarthome* yang dibuat, aplikasi wireshark akan memantau semua lalu lintas paket yang ada di dalam interface jaringan yang ditangkap contoh hasil tangkapan aplikasi wireshark dapat dilihat pada **Gambar 6.2** berikut:



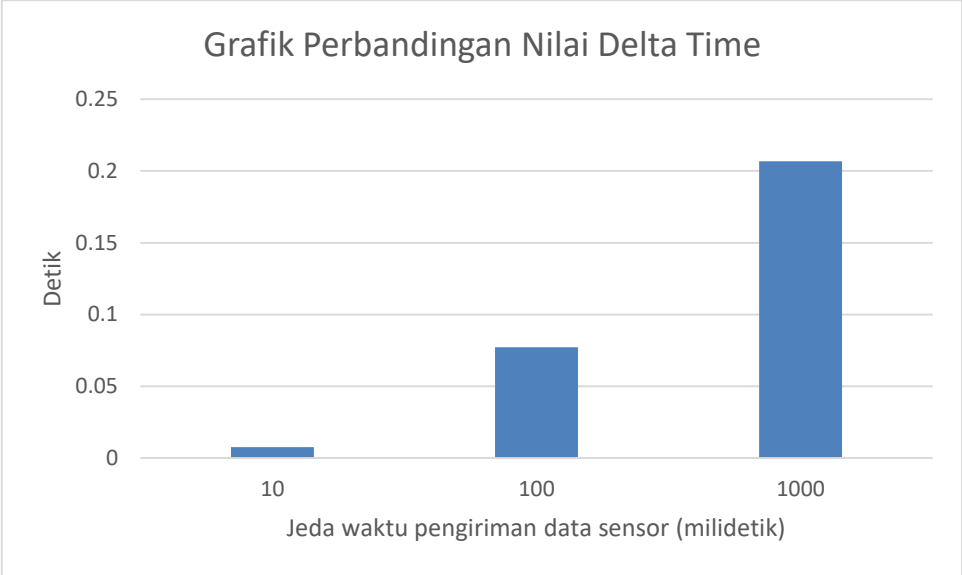
Gambar 6.2 Gambar Hasil Tangkapan Wireshark

Pada **Gambar 6.2** adalah contoh hasil tangkapan aplikasi wireshark yang kemudian di filter menggunakan filter *MQTT* yang berarti aplikasi wireshark hanya menampilkan paket yang dikirimkan menggunakan protokol *MQTT*. Pada proses pengujian aplikasi wireshark akan melakukan capture data jaringan selama 15 menit, dari hasil tangkapan aplikasi wireshark diperoleh hasil dari pengujian yang dapat dilihat pada **Tabel 6.11** berikut:

Tabel 6.11 Hasil Pengujian Kehandalan Sistem

Jeda Pengiriman Data	Delta time	Rata – rata jumlah paket dikirimkan perdetik	Presentase jumlah pengiriman paket perdetik (PPS/packet)*100
10 milidetik	0.007675	36.53333	0.106322%
100 milidetik	0.077292	7.933333	0.111069%
1000 milidetik	0.206807	3.4	0.110884%

Dari **Tabel 6.11** didapatkan beberapa hasil parameter jaringan yaitu, throughput yang merupakan besar data rate yang digunakan saat pengiriman data dalam satuan waktu, delta time adalah interval waktu penerimaan paket dari satu pake ke paket selanjutnya, kemudian presentase jumlah pengiriman paket perdetik, dan presentase besar pengiriman data perdetik *Delta time* dapat direpresentasikan menggunakan grafik seperti pada **Gambar 6.3**.



Gambar 6.3 Grafik Perbandingan Nilai Delta Time

Pada **Gambar 6.3** dapat dilihat perbandingan nilai delta time, nilai tertinggi delta time adalah pada skenario 1000 milidetik, hal ini terjadi karena jeda waktu data yang dikirimkan lebih lama dibandingkan dengan skenario lainnya sehingga mengakibatkan delta time yang tinggi, pada saat skenario 10milidetik mosquitto broker menerima paket dengan sangat cepat dan meneruskan paket yang diterima secara langsung pada node yang dituju, penggunaan method callback dan on_message pada protokol *MQTT* memungkinkan untuk melakukan *publish* data secara langsung ketika node menerima data dari broker, sehingga ketika broker menerima data dengan cepat dan langsung mengirimkannya kepada *subscriber* maka *subscriber* dalam hal ini aplikasi sistem kendali yang memiliki peran sebagai *publisher* dan *subscriber*, secara langsung mem*publish* data ketika menerima data

dari broker, karena itu jeda waktu pengiriman data sangat berpengaruh pada delta time dimana semakin lama jeda pengiriman data maka semakin besar pula delta timenya.

6.2.1.2 Pengujian Integritas Data

Pada pengujian integritas data dilakukan dengan cara membandingkan 50 data sensor pertama yang diterima oleh server dengan 50 data pertama yang di publish oleh masing – masing mikrokontroller yang terintegrasi oleh sensor. Untuk membandingkan data – data tersebut, data yang di terima oleh server disimpan pada database, dan data yang di publish mikrokontroller akan di *capture* oleh aplikasi wireshark. Hasil pengujian dengan skenario pertama yaitu dengan jeda pengiriman data sensor setiap 1000ms dapat dilihat pada **Tabel 6.12** dan **Tabel 6.13** berikut:

Tabel 6.12 Perbandingan Data Sensor Data Suhu yang Diterima dengan yang Dipublish pada jeda pengiriman 1000ms

Sensor suhu jeda pengiriman data 1000 ms										
Data Sensor Suhu	**Data 1**	**Data 2**	**Data 3**	**Data 4**	**Data 5**	**Data 6**	**Data 7**	**Data 8**	**Data 9**	**Data 10**
redis	31	31	26	26	27	27	27	27	27	27
mikrokontroler	31	31	26	26	27	27	27	27	27	27
Data Sensor Suhu	**Data 11**	**Data 12**	**Data 13**	**Data 14**	**Data 15**	**Data 16**	**Data 17**	**Data 18**	**Data 19**	**Data 20**
redis	27	27	27	27	21	21	26	26	27	27
mikrokontroler	27	27	27	27	21	21	26	26	27	27
Data Sensor Suhu	**Data 21**	**Data 22**	**Data 23**	**Data 24**	**Data 25**	**Data 26**	**Data 27**	**Data 28**	**Data 29**	**Data 30**
redis	28	28	27	27	27	27	26	26	26	26
mikrokontroler	28	28	27	27	27	27	26	26	26	26
Data Sensor Suhu	**Data 31**	**Data 32**	**Data 33**	**Data 34**	**Data 35**	**Data 36**	**Data 37**	**Data 38**	**Data 39**	**Data 40**
redis	26	26	26	26	26	26	26	26	26	26
mikrokontroler	26	26	26	26	26	26	26	26	26	26
Data Sensor Suhu	**Data 41**	**Data 42**	**Data 43**	**Data 44**	**Data 45**	**Data 46**	**Data 47**	**Data 48**	**Data 49**	**Data 50**
redis	26	26	27	27	26	26	26	26	26	26
mikrokontroler	26	26	27	27	26	26	26	26	26	26

Tabel 6.13 Perbandingan Data Sensor Data Cahaya yang Diterima dengan yang Dipublish pada jeda pengiriman 1000ms

Sensor cahaya jeda pengiriman data 1000 ms										
Data Sensor Cahaya	**Data 1**	**Data 2**	**Data 3**	**Data 4**	**Data 5**	**Data 6**	**Data 7**	**Data 8**	**Data 9**	**Data 10**
redis	82	100	100	100	100	100	100	100	100	100
mikrokontroler	82	100	100	100	100	100	100	100	100	100
Data Sensor Cahaya	**Data 11**	**Data 12**	**Data 13**	**Data 14**	**Data 15**	**Data 16**	**Data 17**	**Data 18**	**Data 19**	**Data 20**

redis	100	100	100	100	100	100	100	100	100	100
mikrokontroler	100	100	100	100	100	100	100	100	100	100
Data Sensor Cahaya	**Data 21**	**Data 22**	**Data 23**	**Data 24**	**Data 25**	**Data 26**	**Data 27**	**Data 28**	**Data 29**	**Data 30**
redis	100	100	100	100	100	100	100	100	100	100
mikrokontroler	100	100	100	100	100	100	100	100	100	100
Data Sensor Cahaya	**Data 31**	**Data 32**	**Data 33**	**Data 34**	**Data 35**	**Data 36**	**Data 37**	**Data 38**	**Data 39**	**Data 40**
redis	100	100	100	100	100	100	100	100	100	100
mikrokontroler	100	100	100	100	100	100	100	100	100	100
Data Sensor Cahaya	**Data 41**	**Data 42**	**Data 43**	**Data 44**	**Data 45**	**Data 46**	**Data 47**	**Data 48**	**Data 49**	**Data 50**
redis	100	100	100	100	100	100	100	100	100	100
mikrokontroler	100	100	100	100	100	100	100	100	100	100

Pada **Tabel 6.12** dan **Tabel 6.13** di tampilkan 50 data pertama dari masing - masing data sensor yang tersimpan pada database redis dan 50 data pertama dari masing – masing data sensor yang ter *capture* oleh aplikasi wireshark, dari kedua diatas dapat dilihat tidak ada perbedaan data yang tersimpan maupun data yang ter*capture* ini menunjukan integritas data pada jeda pengiriman setiap 1000ms adalah 100%. Hasil pengujian dengan skenario pertama yaitu dengan jeda pengiriman data sensor setiap 100ms dapat dilihat pada **Tabel 6.14** dan **Tabel 6.15** berikut:

Tabel 6.14 Perbandingan Data Sensor Data Suhu yang Diterima dengan yang Dipublish pada jeda pengiriman 100ms

Sensor suhu jeda pengiriman data 100 ms										
Data Sensor Suhu	**Data 1**	**Data 2**	**Data 3**	**Data 4**	**Data 5**	**Data 6**	**Data 7**	**Data 8**	**Data 9**	**Data 10**
redis	23	23	23	23	23	23	23	23	23	23
mikrokontroler	23	23	23	23	23	23	23	23	23	23
Data Sensor Suhu	**Data 11**	**Data 12**	**Data 13**	**Data 14**	**Data 15**	**Data 16**	**Data 17**	**Data 18**	**Data 19**	**Data 20**
redis	23	23	33	33	33	33	33	33	33	33
mikrokontroler	23	23	33	33	33	33	33	33	33	33
Data Sensor Suhu	**Data 21**	**Data 22**	**Data 23**	**Data 24**	**Data 25**	**Data 26**	**Data 27**	**Data 28**	**Data 29**	**Data 30**
redis	33	33	33	33	24	24	24	24	24	24
mikrokontroler	33	33	33	33	24	24	24	24	24	24
Data Sensor Suhu	**Data 31**	**Data 32**	**Data 33**	**Data 34**	**Data 35**	**Data 36**	**Data 37**	**Data 38**	**Data 39**	**Data 40**
redis	24	24	24	24	24	24	24	24	24	24
mikrokontroler	24	24	24	24	24	24	24	24	24	24
Data Sensor Suhu	**Data 41**	**Data 42**	**Data 43**	**Data 44**	**Data 45**	**Data 46**	**Data 47**	**Data 48**	**Data 49**	**Data 50**
redis	24	24	24	24	24	24	24	24	24	24
mikrokontroler	24	24	24	24	24	24	24	24	24	24

Tabel 6.15 Perbandingan Data Sensor Data Cahaya yang Diterima dengan yang Dipublish pada jeda pengiriman 100ms

Sensor cahaya jeda pengiriman data 100 ms										
Data Sensor Cahaya	**Data 1**	**Data 2**	**Data 3**	**Data 4**	**Data 5**	**Data 6**	**Data 7**	**Data 8**	**Data 9**	**Data 10**
redis	88	84	84	85	85	85	85	85	83	87
mikrokontroler	88	84	84	85	85	85	85	85	83	87
Data Sensor Cahaya	**Data 11**	**Data 12**	**Data 13**	**Data 14**	**Data 15**	**Data 16**	**Data 17**	**Data 18**	**Data 19**	**Data 20**
redis	90	88	93	91	89	94	92	95	94	94
mikrokontroler	90	88	93	91	89	94	92	95	94	94
Data Sensor Cahaya	**Data 21**	**Data 22**	**Data 23**	**Data 24**	**Data 25**	**Data 26**	**Data 27**	**Data 28**	**Data 29**	**Data 30**
redis	91	91	93	94	92	94	91	91	94	92
mikrokontroler	91	91	93	94	92	94	91	91	94	92
Data Sensor Cahaya	**Data 31**	**Data 32**	**Data 33**	**Data 34**	**Data 35**	**Data 36**	**Data 37**	**Data 38**	**Data 39**	**Data 40**
redis	94	91	91	94	92	91	91	89	92	90
mikrokontroler	91	91	89	94	92	91	91	89	92	90
Data Sensor Cahaya	**Data 41**	**Data 42**	**Data 43**	**Data 44**	**Data 45**	**Data 46**	**Data 47**	**Data 48**	**Data 49**	**Data 50**
redis	91	93	90	89	92	92	90	87	78	80
mikrokontroler	91	93	90	89	92	92	90	87	78	80

Pada **Tabel 6.14** dan **Tabel 6.15** di tampilkan 50 data pertama dari masing - masing data sensor yang tersimpan pada database redis dan 50 data pertama dari masing – masing data sensor yang ter *capture* oleh aplikasi wireshark, dari kedua diatas dapat dilihat tidak ada perbedaan data yang tersimpan maupun data yang ter*capture* ini menunjukan integritas data pada jeda pengiriman setiap 100ms adalah 100%. Hasil pengujian dengan skenario pertama yaitu dengan jeda pengiriman data sensor setiap 10ms dapat dilihat pada **Tabel 6.17** dan **Tabel 6.18** berikut:

Tabel 6.16 Perbandingan Data Sensor Data Suhu yang Diterima dengan yang Dipublish pada jeda pengiriman 10ms

Sensor suhu jeda pengiriman data 10 ms										
Data Sensor Suhu	**Data 1**	**Data 2**	**Data 3**	**Data 4**	**Data 5**	**Data 6**	**Data 7**	**Data 8**	**Data 9**	**Data 10**
redis	26	26	26	26	26	26	26	26	26	26
mikrokontroler	26	26	26	26	26	26	26	26	26	26
Data Sensor Suhu	**Data 11**	**Data 12**	**Data 13**	**Data 14**	**Data 15**	**Data 16**	**Data 17**	**Data 18**	**Data 19**	**Data 20**
redis	26	26	26	26	26	26	26	26	26	26
mikrokontroler	26	26	26	26	26	26	26	26	26	26
Data Sensor Suhu	**Data 21**	**Data 22**	**Data 23**	**Data 24**	**Data 25**	**Data 26**	**Data 27**	**Data 28**	**Data 29**	**Data 30**
redis	26	26	26	26	26	26	26	26	32	32

mikrokontroler	26	26	26	26	26	26	26	26	32	32
Data Sensor Suhu	**Data 31**	**Data 32**	**Data 33**	**Data 34**	**Data 35**	**Data 36**	**Data 37**	**Data 38**	**Data 39**	**Data 40**
redis	32	32	32	32	32	32	32	32	32	32
mikrokontroler	32	32	32	32	32	32	32	32	32	32
Data Sensor Suhu	**Data 41**	**Data 42**	**Data 43**	**Data 44**	**Data 45**	**Data 46**	**Data 47**	**Data 48**	**Data 49**	**Data 50**
Redis	32	32	32	32	32	32	32	32	32	32
mikrokontroler	32	32	32	32	32	32	32	32	32	32

Tabel 6.17 Perbandingan Data Sensor Data Cahaya yang Diterima dengan yang Dipublish pada jeda pengiriman 10ms

Sensor cahaya jeda pengiriman data 10 ms										
Data Sensor Cahaya	**Data 1**	**Data 2**	**Data 3**	**Data 4**	**Data 5**	**Data 6**	**Data 7**	**Data 8**	**Data 9**	**Data 10**
Redis	66	62	66	62	65	66	65	63	65	66
mikrokontroler	66	62	66	62	65	66	65	63	65	66
Data Sensor Cahaya	**Data 11**	**Data 12**	**Data 13**	**Data 14**	**Data 15**	**Data 16**	**Data 17**	**Data 18**	**Data 19**	**Data 20**
Redis	63	67	66	63	67	66	64	67	64	67
mikrokontroler	63	67	66	63	67	66	64	67	64	67
Data Sensor Cahaya	**Data 21**	**Data 22**	**Data 23**	**Data 24**	**Data 25**	**Data 26**	**Data 27**	**Data 28**	**Data 29**	**Data 30**
Redis	66	63	64	67	64	65	67	66	63	67
mikrokontroler	66	63	64	67	64	65	67	66	63	67
Data Sensor Cahaya	**Data 31**	**Data 32**	**Data 33**	**Data 34**	**Data 35**	**Data 36**	**Data 37**	**Data 38**	**Data 39**	**Data 40**
Redis	66	63	68	66	64	66	64	65	67	64
mikrokontroler	66	63	68	66	64	66	64	65	67	64
Data Sensor Cahaya	**Data 41**	**Data 42**	**Data 43**	**Data 44**	**Data 45**	**Data 46**	**Data 47**	**Data 48**	**Data 49**	**Data 50**
Redis	66	68	66	63	65	68	66	64	67	67
mikrokontroler	66	68	66	63	65	68	66	64	67	67

Pada **Tabel 6.17** dan **Tabel 6.18** di tampilkan 50 data pertama dari masing - masing data sensor yang tersimpan pada database redis dan 50 data pertama dari masing – masing data sensor yang ter capture oleh aplikasi wireshark, dari kedua diatas dapat dilihat tidak ada perbedaan data yang tersimpan maupun data yang tercapture ini menunjukan integritas data pada jeda pengiriman setiap 10ms adalah 100%. Dari hasil pengujian integritas data dengan menggunakan 3 skenario yang berbeda seperti yang telah dijelaskan di atas mendapatkan hasil 100% kesamaan data yang dikirim dengan data yang diterima. Ini menunjukan protokol

MQTT pada penelitian ini merupakan protokol yang reliabel untuk pengiriman data.

6.2.2 Hasil dan Analisis Pengujian Fungsionalitas Aplikasi Sistem Kendali

Hasil dan analisis pengujian fungsionalitas aplikasi sistem kendali dilakukan untuk melihat kesesuaian fungsi – fungsi hasil implementasi dengan perancangan, dari hasil analisis pengujian fungsionalitas dapat dilihat apakah fungsi yang ada pada aplikasi sistem kendali telah berjalan dengan benar, hasil dan analisi pengujian fungsionalitas aplikasi sistem kendali dapat dilihat pada **Tabel 6.18** berikut:

Tabel 6.18 Hasil dan Analisis Pengujian Fungsionalitas Aplikasi Sistem Kendali.

No	Deskripsi	Masukan	Keluaran yang Diharapkan	Hasil yang Didapat	Kesimpulan
1.	Pengujian Menyalakan dan mematikan lampu LED secara otomatis sesuai dengan parameter ketika aplikasi dijalankan	Aplikasi dijalankan	• Aplikasi Menunjukan state/kondisi dari lampu LED 1 dan lampu LED 2 • Kondisi lampu LED 1 dan LED 2 sesuai dengan data sensor yang diterima	• Aplikasi Menunjukan state/kondisi dari lampu LED 1 dan lampu LED 2 • Kondisi lampu LED 1 dan LED 2 sesuai dengan data sensor yang diterima	Benar
2.	Pengujian Menyalakan lampu LED 1 dengan menekan tombol ON yang sejajar dengan label LED 1.	Tekan tombol ON yang berada sejajar dengan label LED 1.	• Aplikasi Menunjukan state/kondisi dari lampu LED 1 dengan status ON • Lampu LED 1 menyala	• Aplikasi Menunjukan state/kondisi dari lampu LED 1 dengan status ON • Lampu LED 1 menyala.	Benar
3.	Pengujian Menyalakan lampu LED 2 dengan menekan tombol ON yang sejajar dengan label LED 2.	Pengguna menekan tombol ON yang berada sejajar dengan label LED 2.	• Aplikasi Menunjukan state/kondisi dari lampu LED 2 dengan status ON • Lampu LED 2 menyala.	• Aplikasi Menunjukan state/kondisi dari lampu LED 2 dengan status ON • Lampu LED 2 menyala.	Benar
4.	Pengujian Mematikan	Pengguna menekan	• Aplikasi Menunjukan	• Aplikasi Menunjukan	Benar

	lampu LED 1 dengan menekan tombol OFF yang sejajar dengan label LED 1	tombol OFF yang berada sejajar dengan label LED 1	state/kondisi dari lampu LED 1 dengan status OFF • Lampu LED 1 mati.	state/kondisi dari lampu LED 1 dengan status OFF • Lampu LED 1 mati.	
5.	Pengujian Mematikan lampu LED 2 dengan menekan tombol OFF yang sejajar dengan label LED 2	Pengguna menekan tombol OFF yang berada sejajar dengan label LED 2	• Aplikasi Menunjukan state/kondisi dari lampu LED 2 dengan status OFF • Lampu LED 2 mati	• Aplikasi Menunjukan state/kondisi dari lampu LED 2 dengan status OFF • Lampu LED 2 mati	Benar
6.	Pengujian menyalakan dan mematikan lampu LED 1 secara otomatis dengan menekan tombol auto yang sejajar dengan label LED 1	Pengguna menekan tombol auto yang berada sejajar dengan label LED 1	• Aplikasi Menunjukan state/kondisi dari lampu LED 1. • State atau kondisi LED 1 sesuai dengan data sensor suhu yang diterima.	• Aplikasi Menunjukan state/kondisi dari lampu LED 1. • State atau kondisi LED 1 sesuai dengan data sensor suhu yang diterima.	Benar
7.	Pengujian menyalakan dan mematikan lampu LED 2 secara otomatis dengan menekan tombol auto yang sejajar dengan label LED 2.	Pengguna menekan tombol auto yang berada sejajar dengan label LED 2.	• Aplikasi Menunjukan state/kondisi dari lampu LED 2. • State atau kondisi LED 2 sesuai dengan data sensor cahaya yang diterima.	• Aplikasi Menunjukan state/kondisi dari lampu LED 2. • State atau kondisi LED 2 sesuai dengan data sensor cahaya yang diterima.	Benar
8.	Pengujian menampilkan grafik pada aplikasi	Aplikasi dijalankan	• Aplikasi menampilkan grafik sesuai dengan nilai pada database redis	• Aplikasi menampilkan grafik sesuai dengan nilai pada database redis	Benar

9.	Pengujian merubah nilai parameter suhu dan cahaya	Pengguna melakukan input pada input box yang telah disediakan kemudian menekan tombol Enter	• Aplikasi merubah nilai parameter suhu dan cahaya sesuai dengan nilai yang diinputkan pengguna.	• Aplikasi merubah nilai parameter suhu dan cahaya sesuai dengan nilai yang diinputkan pengguna.	Benar
10	Pengujian Menampilkan Nilai data sensor pada label diaplikasi	Aplikasi di jalankan	• Aplikasi menampilkan nilai terakhir dari data sensor suhu dan data sensor cahaya yang diterima	• Aplikasi menampilkan nilai terakhir dari data sensor suhu dan data sensor cahaya yang diterima	Benar

Dari **Tabel 6.18** dapat dilihat hasil dari pengujian fungsional dimana setiap fungsi berfungsi dengan benar sesuai dengan output yang diharapkan, selain itu keberhasilan dari fungsi aplikasi juga menggambarkan bahwa sistem yang dibangun dapat terhubung dan terintegerasi dengan baik. Terintegerasinya sistem ditunjukan pada baris no 1-7 dari **Tabel 6.18** dimana input yang masuk pada aplikasi sistem kendali menentukan kondisi atau state dari lampu LED 1 dan LED 2 ini berarti setiap node dapat saling berkomunikasi dan aplikasi sistem kendali berhasil melakukan fungsi monitoring data sensor dan kontroling lampu LED 1 dan LED 2, terintegerasinya sistem juga di karenakan semua node terhubung pada akses point yang sama.

Keberhasilan terintegerasinya sistem juga menunjukan keberhasilan dari implementasi protokol *MQTT*, di karenakan komunikasi yang terjadi pada sistem untuk pengiriman data menggunakan protokol *MQTT* selain itu keberhahasilan protokol *MQTT* juga dapat terlihat pada sesuainya kondisi dari lampu LED dengan input yang diterima oleh aplikasi sistem kendali baik itu input yang diperoleh dari hasil *publish* data sensor oleh mikrokontroler maupun inputan dari trigger *button* pada aplikasi, karena state dari lampu LED 1 dan LED 2 akan berubah sesuai topik yang di *subscribe* dimana telah dijelaskan pada bab implementasi bahwa LED 1 dan LED 2 men*subscribe* topik yang berbeda, begitu pula pada aplikasi sistem kendali yang melakukan *subscribe* pada 2 topik berbeda ketika menerima inputan dengan topik yang sama dengan yang di *subscribe* oleh aplikasi, maka aplikasi akan mem*publish* pesan sesuai dengan inputan yang diterima.

Keberhasilan pengimplementasian protokol *MQTT* tidak terlepas dari peran mosquitto broker sebagai perantara dan penerus semua pesan, mikrokontroler yang terintegerasi dengan sensor suhu dan mikrokontroler yang terintegrasi dengan sensor cahaya mem*publish* data sensor kepada broker, pada **Tabel 6.18** dapat dilihat keberhasilan aplikasi sistem kendali menampilkan nilai data sensor

yang diterima ini berarti aplikasi dapat dengan baik menerima data dari broker, begitu pula ketika berubahnya state/kondisi dari lampu LED menunjukan mikrokontroler yang terintegerasi dengan lampu LED 1 dan LED 2 dapat menerima pesan dari broker dengan baik sesuai dengan topik yang di *subscribe*, karena pesan yang diterima oleh mikrokontroler dengan lampu LED adalah pesan yang di*publish* oleh aplikasi sistem kendali pada broker.

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, pengujian, serta analisis dari Sistem Kendali Berbasis Mikrokontroler Menggunakan Protokol *MQTT* pada *Smarthome* dapat disimpulkan bahwa:

1. Terintegerasinya sistem dilihat dari berjalannya aplikasi sistem kendali sesuai dengan keluaran yang diharapkan yaitu dapat melakukan monitoring data sensor dan kontroling terhadap lampu LED 1 dan lampu LED 2.
2. Penerapan protokol *MQTT* untuk komunikasi di dalam sistem *smarthome* digunakan untuk pengiriman pesan pada sistem dengan mekanisme *publish*/*subscribe*, dimana *publisher* akan mengirim pesan dengan topik tertentu dan *subscriber* akan menerima pesan sesuai dengan topik yang di*subscribe*, sehingga pengiriman dan penerimaan pesan dapat sesuai berdasarkan topik yang ditentukan.
3. Mekanisme pengiriman antar node didalam sistem menggunakan mekanisme *publish*/*subscribe* dari *MQTT* dimana mikrokontroler yang terintegerasi dengan sensor melakukan *publish* pesan dengan topik yang telah ditentukan pada broker, kemudian aplikasi sistem kendali menerima data dari broker sesua dengan topik yang di*subscribe* oleh aplikasi, aplikasi kemudian akan mem*publish* pesan pada broker sesuai dengan input yang diterima oleh aplikasi, pesan yang di *publish* oleh aplikasi akan diteruskan oleh broker pada mikrokontroler yang terintegerasi dengan LED untuk menentukan state/kondisi dari lampu LED 1 dan lampu LED 2.
4. Berdasarkan hasil pengujian kehandalan sistem banyaknya paket yang dikirimkan menggunakan protokol *MQTT* dalam satu waktu mempengaruhi nilai delta time dimana semakin singkat jeda pengiriman waktu semakin kecil nilai delta timenya, dan nilai integritas data yang dikirm dan diterima melalui protokol *MQTT* adalah 100%.

7.2 Saran

Setelah menyelesaikan penelitian ada beberapa saran yang dapat disampaikan oleh penulis guna untuk mengembangkan Aplikasi Sistem Kendali Berbasis Mikrokontroler Menggunakan Protokol *MQTT pada Smarthome*:

1. Perlu dilakukan percobaan dengan jumlah sensor yang lebih banyak dan jenis sensor yang lebih beragam.
2. Perlu dilkakuan penelitian dengan menggunakan jenis mikrokontroler yang berbeda selain menggunakan mikrontroler wemos D1 R2.

3. Perlu dilakukan pengimplementasian yang lebih beragam selain menggunakan lampu LED untuk merepresentasikan device pada *smarthome*.

DAFTAR PUSTAKA

Adi, H. K., Sakti, E. & Amron, K., 2016. *PENGUMPULAN DATA MENGGUNAKAN METODE PUBLISH SUBSCRIBE PADA NODE SENSOR DALAM WIRELESS MESH NETWORK,* MALANG: s.n.

Aroon, N., 2016. Study of using *MQTT* Cloud Platform for Remotely.

Eclipse, 2011. *Eclipse Paho.* [Online] Available at: https://eclipse.org/paho/ [Accessed 30 12 2016].

Eclipse, 2013. *Mosquitto.* [Online] Available at: https://www.eclipse.org/proposals/technology.mosquitto/ [Accessed 30 12 2016].

Kim, S.-M., Choi, H.-S. & Rhee, W.-S., 2015. *IoT* Home Gateway for Auto-Configuration and.

Klein, J., Gorton, I. & Pham, K., 2015. Application-Specific Evaluation of NoSQL Databases. *IEEE International Congress on Big Data.*

Lampkin, V. et al., 2012. *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry.* 1st ed. USA: WebSphere MQ.

Mohammad, A., Abedin, A. & Rahman Khan, Z., 2009. Microcontroller Based Control System for Electric.

Naglič, M. & Souvent, A., 2013. Concept of *Smarthome* and SmartGrids integration.

Nusantara, M. F., Akbar, S. R. & Rachmadi, A., 2016. Analisa Metode Publish/subcribe untuk komunikasi data antar perangkat dalam lingkungan *smarthome*.

Solace, n.d. *MQTT Topics.* [Online] Available at: http://docs.solace.com/Features/*MQTT*-Topics.htm [Accessed 30 12 2016].

sparkfun, n.d. *WiFi Module - ESP8266.* [Online] Available at: https://www.sparkfun.com/products/13678 [Accessed 30 12 2016].

Thangave, D., Ma, X. & TAN, C. K.-Y., 2014. Performance Evaluation of *MQTT* and CoAP.

Upadhyay, Y., Borole, M. & Mr.D.Dileepan, 2016. *MQTT* Based Secured Home Automation System. p. 4.

Wemos, n.d. *Wemos D1.* [Online] Available at: https://www.wemos.cc/product/d1.html [Accessed 30 12 2016].

LAMPIRAN KODE PROGRAM

A.1 Mikrokontroler+DHT11

```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <PubSubClient.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
#define DHTTYPE DHT11   // DHT 11
#define D0 16
#define D1 5 // I2C Bus SCL (clock)
#define D2 4 // I2C Bus SDA (data)
#define D3 0
#define D4 2 // Same as “LED_BUILTIN”, but inverted logic
#define D5 14 // SPI Bus SCK (clock)
#define D6 12 // SPI Bus MISO
#define D7 13 // SPI Bus MOSI
#define D8 15 // SPI Bus SS (CS)
#define D9 3 // RX0 (Serial console)
#define D10 1 // TX0 (Serial console)
#define DHTPIN D5
DHT dht(DHTPIN, DHTTYPE);
const char *ssid = "hudan";
const char *password = "12345678";
IPAddress ip(192,168,0,99);
IPAddress gateway(192,168,0,1);
IPAddress subnet(255,255,255,0);
const char* MQTT_server = "192.168.0.100";
char temp[50];
String temp_str;
```

```
WiFiClient espClient;
PubSubClient client(espClient);
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  setup_wifi();
  Serial.println("DHT test!");
  dht.begin();
  Serial.println("Booting");

  client.setServer(MQTT_server, 1883);
  //client.setCallback(callback);
  WiFi.printDiag(Serial);
}
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_AP);
  WiFi.begin(ssid, password);
  WiFi.config(ip,gateway,subnet);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    Serial.print(".");
    ESP.restart();
  }}
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
```

```
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("DHT")) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  //float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();



  // Check if any reads failed and exit early (to try again).
  if (isnan(t) ) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // Compute heat index in Celsius (isFahreheit = false)
```

```
//  float hic = dht.computeHeatIndex(t, h, false);
  temp_str = String(t); //converting ftemp (the float variable above) to a string
  temp_str.toCharArray(temp, temp_str.length() + 1); //packaging up the data to publish to MQTT
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");

  Serial.print(" %\t");

  //Serial.print("Heat index: ");
  //Serial.print(hic);
  //Serial.print(" *C ");

   Serial.println();
   Serial.print("Publish message: ");
   Serial.println(t);
   client.publish("suhu",temp);

  delay(10);

  }
```

A.2 Mikrokontroler+LDR

```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <PubSubClient.h>
#include <RBD_LightSensor.h>
#define DHTTYPE DHT11   // DHT 11
```

```
#define D0 16
#define D1 5 // I2C Bus SCL (clock)
#define D2 4 // I2C Bus SDA (data)
#define D3 0
#define D4 2 // Same as “LED_BUILTIN”, but inverted logic
#define D5 14 // SPI Bus SCK (clock)
#define D6 12 // SPI Bus MISO
#define D7 13 // SPI Bus MOSI
#define D8 15 // SPI Bus SS (CS)
#define D9 3 // RX0 (Serial console)
#define D10 1 // TX0 (Serial console)
IPAddress ip(192,168,0,112);
IPAddress gateway(192,168,0,1);
IPAddress subnet(255,255,255,0);
// Update these with values suitable for your network.
const char* ssid = "hudan";
const char* password = "12345678";
const char* MQTT_server = "192.168.0.100";
float l;
char light[50];
String light_str;
WiFiClient espClient;
PubSubClient client(espClient);
RBD::LightSensor light_sensor(A0);
void setup() {
  Serial.begin(115200);
  setup_wifi();
  Serial.println("Booting");
  client.setServer(MQTT_server, 1883);
}
void setup_wifi() {
  delay(10);
```

```
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_AP);
  WiFi.begin(ssid, password);
  WiFi.config(ip,gateway,subnet);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
    Serial.print(".");
    ESP.restart();
  }

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("LDR")) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
void loop() {
   ArduinoOTA.handle();
```

```
  delay(200);
 if (!client.connected()) {
   reconnect();
 }
 client.loop();
 Serial.println(light_sensor.getPercentValue());
 l=light_sensor.getPercentValue();
 light_str = String(l);
 light_str.toCharArray(light, light_str.length() + 1); //packaging up the data to publish to MQTT
 client.publish("cahaya",light);
 delay(1000);
 }
```

A.3 Mikrokontroler+LED

```
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <PubSubClient.h>
#define D0 16
#define D1 5 // I2C Bus SCL (clock)
#define D2 4 // I2C Bus SDA (data)
#define D3 0
#define D4 2 // Same as “LED_BUILTIN”, but inverted logic
#define D5 14 // SPI Bus SCK (clock)
#define D6 12 // SPI Bus MISO
#define D7 13 // SPI Bus MOSI
#define D8 15 // SPI Bus SS (CS)
#define D9 3 // RX0 (Serial console)
#define D10 1 // TX0 (Serial console)
```

```
IPAddress ip(192,168,0,113);
IPAddress gateway(192,168,0,1);
IPAddress subnet(255,255,255,0);
// Update these with values suitable for your network.
const char* ssid = "hudan";
const char* password = "12345678";
const char* MQTT_server = "192.168.0.100";
WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
  Serial.begin(115200);
  setup_wifi();
  Serial.println("Booting");
  pinMode(D5,OUTPUT);
  pinMode(D7,OUTPUT);
  client.setServer(MQTT_server, 1883);
  client.setCallback(callback);
}

void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_AP);
  WiFi.begin(ssid, password);
  WiFi.config(ip,gateway,subnet);
  while (WiFi.waitForConnectResult() != WL_CONNECTED) {
    Serial.println("Connection Failed! Rebooting...");
    delay(5000);
```

```
    Serial.print(".");
    ESP.restart();
  }
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("LED")) {
      Serial.println("connected");
      client.subscribe("1");
      client.subscribe("2");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  }
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
```

```
  Serial.print((char)payload[i]);
 }
 Serial.println();

 // Switch on the LED if an 1 was received as first character
 if (strcmp(topic,"1")==0) {
  if((char)payload[0] == '1'){
  digitalWrite(D5, LOW);   // Turn the LED on (Note that LOW is the voltage level
  Serial.println("nyala LED 1");
  }
  else{
    digitalWrite(D5, HIGH);
   Serial.println("mati LED 1");
   }
  // but actually the LED is on; this is because
  // it is acive low on the ESP-01)
 }
 if(strcmp(topic,"2")==0) {
    if((char)payload[0] == '1')
    {
  digitalWrite(D7, LOW);
  Serial.println("nyala LED 2");}
    else{
   digitalWrite(D7, HIGH);
   Serial.println("mati LED 2");
      }
 }
 }
```

A.4 Aplikasi Sistem Kendali Python

```
#import lib client paho MQTT
from Tkinter import *
```

```
from ttk import *
from datetime import datetime
import matplotlib
import paho.MQTT.client as MQTT
import redis, time
from matplotlib import style
import matplotlib.pyplot as plt
MQTTc = MQTT.Client("serverclient",clean_session=False)#inisialisasi MQTT client
r = redis.Redis("192.168.0.100",6379)
start = time.time()
date = datetime.now().strftime('%S')
#f = Figure(figsize=(4,3), dpi=80)
style.use('ggplot')
matplotlib.use("TkAgg")
paramsuhu=30
paramcahaya=50
LED1 ='-'
LED2='-'
param="default"
#param=MQTTx.on_message()
def MQTTx():
   #fungsi callback
   def on_message(MQTTc,obj,msg):
      global LED1
      global LED2
      global param
      datasuhu = r.lrange("suhu",-1,-1)
      datacahaya = r.lrange("cahaya",-1,-1)
     # print "Telah Diterima message : "+msg.payload+" topik "+msg.topic
      r.rpush(msg.topic,msg.payload)
      if param=="default":
```

```
        if msg.topic=="suhu":
            if float(msg.payload)<=paramsuhu :
                MQTTc.publish("1","1",qos=0,retain=False)
                LED1="ON"
            elif float(msg.payload)>paramsuhu:
                MQTTc.publish("1","0",qos=0,retain=False)
                LED1="OFF"
        elif msg.topic=="cahaya" :
            if float(msg.payload) <=paramcahaya:
                MQTTc.publish("2","1",qos=0,retain=False)
                LED2="ON"
            elif float(msg.payload)>paramcahaya:
                MQTTc.publish("2","0",qos=0,retain=False)
                LED2="OFF"
    elif param =="auto1":
        if msg.topic=="suhu":
            if float(msg.payload)<=paramsuhu :
                MQTTc.publish("1","1",qos=0,retain=False)
                LED1="ON"
            elif float(msg.payload)>paramsuhu:
                MQTTc.publish("1","0",qos=0,retain=False)
                LED1="OFF"

    elif param=="auto2":
        if msg.topic=="cahaya" :
            if float(msg.payload) <=paramcahaya:
                MQTTc.publish("2","1",qos=0,retain=False)
                LED2="ON"
            elif float(msg.payload)>paramcahaya:
                MQTTc.publish("2","0",qos=0,retain=False)
                LED2="OFF"
    else :
```

```
        print
#registrasi fungsi callback
   MQTTc.on_message =  on_message
#koneksi ke broker
   MQTTc.connect("192.168.0.100",1883)
#subcribe ke topik
   MQTTc.subscribe("suhu")
   MQTTc.subscribe("cahaya")
class Example(Frame):
   def __init__(self, parent):
      Frame.__init__(self, parent)
      self.pack(expand=True, fill='both')
      self.initUI()
      self.CheckVar1=StringVar()
      self.CheckVar2=StringVar()
      self.update()
      self.graph()

   def initUI(self):
      self.style = Style()
      self.style.theme_use("default")
      vcmd = (self.register(self.validate),
            '%d', '%i', '%P', '%s', '%S', '%v', '%V', '%W')
      self.L1 =Label(self,text="Parameter suhu")
      self.L1.grid(row=0)
      self.E1 = Entry(self, validate = 'key',width=3 ,validatecommand = vcmd)
      self.E1.insert(END, '0')
      self.E1.grid(row=0,column=1)
      self.L2 =Label(self,text="Parameter cahaya")
      self.L2.grid(row=1)
      self.E2 = Entry(self, validate = 'key',width=3 ,validatecommand = vcmd)
      self.E2.insert(END, '0')
```

```
        self.E2.grid(row=1, column=1)
        self.submit = Button(self, text="Enter", width=15, command=lambda
:self.cb())
        self.submit.grid(row=2)
        self.L3 =Label(self,text="LED 1 : "+LED1)
        self.L3.grid(row=0, column=2)
        self.L4 =Label(self,text="LED 2 : "+LED2)
        self.L4.grid(row=1, column=2)
        self.L5 =Label(self,text="Suhu : "+str( r.lrange("suhu",-1,-1))+"%C")
        self.L5.grid(row=3)
        self.L6 =Label(self,text="Cahaya : "+ str(r.lrange("cahaya",-1,-1))+"%")
        self.L6.grid(row=4)
        self.L7 =Label(self,text="Kontroling")
        self.L7.grid(row=5)
        self.L8 =Label(self,text="LED 1 ")
        self.L8.grid(row=6,column=0)
        self.on1 = Button(self, text="ON", width=5, command=lambda
:self.kontrol("on1"))
        self.off1 = Button(self, text="OFF", width=5, command=lambda
:self.kontrol("off1"))
        self.auto1 = Button(self, text="Auto", width=5, command=lambda
:self.kontrol("auto1"))
        self.on1.grid(row=6,column=1)
        self.off1.grid(row=6,column=2)
        self.auto1.grid(row=6,column=3)


        self.L9 =Label(self,text="LED 2 ")
        self.L9.grid(row=7,column=0)
        self.on2 = Button(self, text="ON", width=5, command=lambda
:self.kontrol("on2"))
        self.off2 = Button(self, text="OFF", width=5, command=lambda
:self.kontrol("off2"))
        self.auto2 = Button(self, text="Auto", width=5, command=lambda
:self.kontrol("auto2"))
```

```
        self.on2.grid(row=7,column=1)
        self.off2.grid(row=7,column=2)
        self.auto2.grid(row=7,column=3)
        self.x_list_1 = []
        self.y_list_1 = []
        self.x_list_2 = []
        self.y_list_2 = []
        r.rpush("suhu",0.0)
        r.rpush("cahaya",0.0)
    def kontrol(self,x):
        global LED1
        global LED2
        global param
        if x=="on1":
            MQTTc.publish("1","1",qos=0,retain=False)
            if param=="auto2" or param=="default":
                param="auto2"
            else :
                param="on1"
            LED1="ON"
        elif x=="off1":
            MQTTc.publish("1","0",qos=0,retain=False)
            if param=="auto2" or param=="default":
                param="auto2"
            else :
                param="off1"
            LED1="OFF"
        elif x=="auto1":
            if param=="auto2" or param == "default":
                param="default"
            else:
                param="auto1"
```

```
    elif x=="on2":
        MQTTc.publish("2","1",qos=0,retain=False)
        if param=="auto1" or param=="default":
            param="auto1"
        else :
            param="on2"
        LED2="ON"
    elif x=="off2":
        MQTTc.publish("2","0",qos=0,retain=False)
        if param=="auto1" or param=="default":
            param="auto1"
        else :
            param="off2"
        LED2="OFF"
    elif x=="auto2":
        if param=="auto1" or param=="default":
            param="default"
        else:
            param="auto2"
def graph(self):
        suhu1 = r.lrange("suhu",-1,-1)
        cahaya1 = r.lrange("cahaya",-1,-1)
        date = datetime.now().second
        date1 = datetime.now().strftime("%d-%m-%Y")
        date2 = datetime.now().strftime("%H:%M:%S")
        join1="".join(suhu1)
        suhugraph=float(join1)
        join2="".join(cahaya1)
        cahayagraph=float(join2)
        plt.ion()
        plt.xlabel("Waktu")
        plt.ylabel("nilai")
```

```
        plt.axis([0, 60, 0, 100])
        if date==00 :
           plt.cla()
           plt.axis((0, 60, 0, 100))
           self.x_list_1 = []
           self.y_list_1 = []
           self.x_list_2 = []
           self.y_list_2 = []
        plt.suptitle("Grafik nilai suhu dan cahaya \n "+date1+"\n"+date2, fontsize=14, fontweight='bold')
        self.x_list_1.append(date)
        self.y_list_1.append(suhugraph)
        self.x_list_2.append(date)
        self.y_list_2.append(cahayagraph)
        plt.legend(['Suhu','Cahaya'])
        plt.plot(self.x_list_1, self.y_list_1, "r-")
        plt.plot(self.x_list_2, self.y_list_2, "b-")
        self.after(100, self.graph)


   def validate(self, action, index, value_if_allowed,
   prior_value, text, validation_type, trigger_type, widget_name):
      if(action=='1'):
        if text in '0123456789.-':
           try:
              float(value_if_allowed)
              return True
           except ValueError:
              return False
        else:
           return False
      else:
        return True
```

```
    def cb(self):
        global paramsuhu
        global paramcahaya
        self.CheckVar1 = self.E1.get()
        self.CheckVar2 = self.E2.get()
        paramsuhu=float(self.CheckVar1)
        paramcahaya=float(self.CheckVar2)
    def update(self):
        #print('[DEBUG] update_text')
        self.L3['text'] = "LED 1 : " + LED1
        self.L4['text'] = "LED 2 : " + LED2
        self.L5['text'] = "Suhu : " +  str(r.lrange("suhu",-1,-1))+"%C"
        self.L6['text'] = "Cahaya : " +  str(r.lrange("cahaya",-1,-1))+"%"
        # run again after 500ms (0.5s)
        self.after(1000, self.update)


def main():
    root = Tk()
    root.geometry("400x400-200+100")
    app = Example(root)
    root.wm_title("Server")
    root.mainloop()
if __name__ == '__main__':
    MQTTx()
    MQTTc.loop_start()
    main()
```